# Fuzzy Control of a Hyperloop Mass Transit System

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Engineering

by

Joseph D. Plantz
B.S.C.S., Wright State University, 2004

2016
Wright State University

Wright State University
GRADUATE SCHOOL

December 10, 2016

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Joseph D. Plantz ENTITLED Fuzzy Control of a Hyperloop Mass Transit System BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Engineering.

_____

Dr. Kuldip S. Rattan
Thesis Director

_____

Dr. Brian D. Rigling
Department Chair

Committee on
Final Examination

_____

Kuldip S. Rattan, Ph.D.

_____

Marian K. Kazimierczuk, Ph.D.

_____

David C. Gross, Ph.D.

_____

Robert E.W. Fyffe, Ph.D.
Vice President for Research and
Dean of the Graduate School

# ABSTRACT

Plantz, Joseph D. . M.S.Egr., Department of Electrical Engineering, Wright State University, 2016. *Fuzzy Control of a Hyperloop Mass Transit System.*

Fuzzy logic control of a Hyperloop is carried out in this thesis. Hyperloop is being described hypothetically as a fifth mode of mass transportation and is a registered trademark of Space Exploration Technologies Corporation (SpaceX). To inspire others to help in its development and make it a reality, the Hyperloop is being explored as open-source technology by SpaceX.

In this thesis a near frictionless track is constructed and is fixed inside a tunnel. External fans are used to produce air pressure inside the tunnel to propel the vehicle down the track. Fuzzy Logic Control is used to stop the vehicle at a desired location. The objective is to stop the vehicle at various end point positions. It is assumed that the vehicle is traveling at or near the desired velocities before the Fuzzy Logic Controller become active. The results show that the Fuzzy Logic Controller is able to effectively stop the vehicle at or near the desired end point positions given a very dynamic and highly non-linear environment.

# Contents

# List of Figures

# Acknowledgment

I would like to take this opportunity to extend my sincere thanks to my wife Kim for encouraging me to go back to school and for all her sacrificed evenings and weekends that she gave me.

Thanks to our four kids who let dad do homework beside them each night at the kitchen table.

Thanks to my father-in-law Max Littlejohn for the use of his woodworking shop and the loan of his knowledge and craftsmanship.

Thanks to my parents for everything they have done to help get me where I am today.

I would like to thank my colleague Richard Metzger (who is a Controls Genius) for all the tutoring.

I would like to thank Dr. Rattan for all his help and advise.

Thanks to Professor Matthew Rickey for his guidance in Fuzzy Logic.

Dedicated to my loving wife Kim

# Introduction

## 1.1 Problem Description

In 2013, Elon Musk (CEO SpaceX) put forth a white paper entitled, “ Hyperloop Alpha ” [1] in which he described a hypothetical fifth mode of mass transportation now known simply as “ Hyperloop ”. This new system would encapsulate a "pod" or vehicle inside a partially evacuated tube. The pod would be suspended inside the tube or tunnel by some means; proposed magna level technoology or through air diffusion. Both proposed ideas produce the desired "air bearing" to create a frictionless effect and give the pod the capabilities of traveling at speeds at or near the sound barrier. Many ideas put forth to date are propelling the project forward to make this new mode of mass transportation a reality. Those ideas seemingly concentrate around building a frictionless/near frictionless track and propelling the vehicle at high speed.

## 1.2 Previous Systems

The use of a pneumatic tube system for transport is nothing new. This idea was first proposed in 1810 by an English inventor named George Medhurst[7]. Medhurst initially suggested a pneumatic tube system for carrying letters and packages and 2 years later his proposal included a tube system large enough to accommodate a railway for carrying freight

and even passengers. Fearing that passengers would not like the idea of being enclosed inside a tube and being exposed to pressures within the tube, Medhurst modified his initial proposal which would use a pneumatic system to pull railway cars down a track. This pneumatic railway would operate using a small tube running next to an ordinary railway track. The smaller tube contained a piston which would pull an attached train car down rails. This system became known as "the atmospheric railway".[7]

"Four atmospheric railways were constructed in the 1840s, before the use of steam locomotives was fully established. Two of them, in Ireland (1844-1854) and France (1847-1860), were short railways on inclines under two miles in length and used pneumatic power only uphill, coasting by gravity in the other direction. The two others, both in England, attempted to use pneumatic power for longer distance operation. The London and Croydon Railway operated atmospherically at its greatest extent from New Cross to Croydon, seven and a half miles. A trial run in September 1845 achieved 70 miles per hour, an astonishing speed for the time." [7]

In the United States in 1870, Alfred Beach developed a pneumatic transport system in New York that could transport people. The tunnel system was a block long and was built in secrecy under City Hall. Dubbed little more than a public attraction, the system moved passengers at 10 miles an hour through the use of a huge rotary blower which produced one hundred thousand cubic feet of air per minute. [8]

By 1893, pneumatic tube systems were in place in cities such as Philadelphia, Boston, Brooklyn, New York, Chicago and St. Louis and were used to carry mail and small packages. Canisters were used that could hold up to 600 letters and traveled an average of 35 miles per hour. [11]

The use of pneumatic mail systems in the United States were built by contractors. The postal services at that time leased/rented the systems for use. "During World War I, the Post Office Department suspended the service to conserve funding for the war effort. After the war, service was only restored in New York and Boston." [12]

Due to increasing populations and the amount of mail this produced, the pneumatic tube systems strained to keep up with the heavy flow. City by City the tube systems were slowly shut down for the practical reasoning that it was more efficient to ship large quantities of mail by truck (that were slower) than by shipping small quantities at a faster rate (tube system). In 1953 the final tube services in New York City were suspended pending a review of the service. It was not reinstated. [13]

## 1.3 Current Hyperloop Work

The name Hyperloop is a registered trademark of Space Exploration Technologies Corporation also known as SpaceX. Hyperloop technology being explored has been explicitly open-sourced by SpaceX.[14]

The idea behind open sourcing is to inspire others to help in the development of the technologies needed to make the Hyperloop a reality. In response, development teams and competitions have sprung up including colleges across the United State. A competition annoucement from SpaceX[15] sparked excitement and a preliminary pod competition in January 2016 has narrowed a field of 120 schools to just 29 college teams, one high school team and one non-student team.[16] The selected teams will compete on a mile long SpaceX Hyperloop test track[17] for the fastest and best overall pod design. The test track for the competition will be located in Hawthorne California, according to SpaceX, and the event is currently scheduled for a three day competition set for January 27-29, 2017.

New startup companies, such as Hyperloop One[18], have begun working on a Hyperloop propulsion system. On May 11, 2016 in Las Vegas Nevada, Hyperloop One demonstrated their propulsion system on a half-mile open air track. The propulsion system powered a 10 foot sled down the half-mile track reaching a speed of 116 miles an hour.[19]

Another startup company, Hyperloop Transportation Technologies (HTT), has been

working on levitating the Hyperloop pod using a technique called, “passive magnetic levitation ”.[20] First developed by the late physicist Richard Post, Post created a levitating track (called the Inductrack) through the use of permanent magnets. This new technique would use permanent magnets attached to the bottom of the pod in a fixed configuration. Coils of insulating wires are attached to the track. As the pod moves across the track, each coil is induced with a current from the permanent magnets on the pod. The induced current flow in the coils in turn generates an electromagnetic field that repels the permanent magnets on the pod which produces levitation.[20]

To date, no known entity has produced what would be a final Hyperloop system. However, with renewed attention focused on pneumatic tube systems, using todays technologies, a fifth mode of mass transportation may be near.

In this thesis a near frictionless track is built, a vehicle is propelled down that track, and the vehicle decelerates through means of Fuzzy Logic Control at desired positions. The track is constructed from plywood strips and has an overall length of 96". Angled aluminum channel fitted with permanet magnetic strips are fastened length ways to the track and provide the cart a straight and smooth surface. A 3D cart is printed and fitted with opposing magnetic strips to that on the track and give the cart the lift that is needed to create an air bearing between the cart and the track. The cart is designed with a keel that fits into the channel of the track and is used to keep the opposing magnetic strips of the cart centered over the strips of the track. The track is then fitted inside a tube that is to help create the controlled environment. DC fans placed at each end of the tube provide air pressure to move and act upon the cart. A laser rangefinder is used for position sensing of the cart and provides feedback to the control loop of the fuzzy logic controller. The attention is focused on stopping the vehicle at its end points through position control. It is assumed that the vehicle traveling down the tunnel has reached a desired velocity before the Fuzzy Controller becomes active.

# System Description

The system described in this chapter will focus around the following requiements:

- Create a near frictionless track encapsulated in a tunnel
- Build a vehicle to travel down that track
- Use DC fans to produce air flow as a means of force to act on the vehicle
- Design a Fuzzy Logic Position Controller to control endpoint positions

## 2.1 System and Setup

The setup for this system can be broken into three catagories with underlining sub-catagories:

- Hardware
  - Track Construction
  - Pod Construction
- Electronic
  - DC Fans
  - Power Supplies
  - Signal Conditioning box

  - Position Sensor - Rangefinder
  - Arduino Mega 2560
- Software
  - MATLAB/Simulink

## 2.2 Hardware

Track and calibration material used for construction of this project were purchased from a local hardware store. Electronics components were purchased from online vendors such as Digi-key[21]. Construction of the preliminary and final tracks were completed in a wood crafting workshop. Plastic tubing for the tunnel was also purchased from a local plastics vendor. At each stage in the devleopment of the track and the system as a whole, many of the decisions that helped evolve into the final system came from fundamental pre-testing of each sub-system. Because of the vast amount of the many hand tools and machines used to construct this project, a detailed list has not been provided. Rather, each system is presented as it was developed.

### 2.2.1 Track Construction

Several preliminary tracks and subsystems were created before constructing the final track. The first preliminary track was built to gain a better understanding of how best to position the magnetic strips and pod placement over those strips. The track was 2 feet in length and fashioned so that the track adjustments could easily be made. This first track was made for observation so size was not a concern at this point. Observing behavior such as pod traveled, the effects and behavior of different sized keel's were made and noted for future work. The top of the track was fitted with bolts to allow repositioning. See appendix A.0.1 for the first preliminary track.

The second preliminary track was constructed to fit inside of a 4" tube. The track was made from a 2"x4"x10' wooden board with a channel cut through the middle to provide space for the keel of the pod. See appendix A.0.2. The track was fitted into the tube and a DC fans was attached to the end. See appendix A.0.3. Preliminary testing began with open loop voltage being applied to the fan to propel the pod down the track. See appendix A.0.4.

Final track construction began with a 4'x8' sheet of $\frac{3}{4}$ " plywood. The plywood was cut lengthways into 6" strips. Four of the strips were glued on each side, stacked on top of one another and then clamped for drying. After the glue was set, a 2-$\frac{1}{2}$ " x $\frac{1}{4}$ " channel was cut through the middle of the stack plywood sheets. This channel is used by the keel of the pod and keeps the pod resting on top of the magnetic strips. Figure 2.1 shows the layered plywood track with the keel channel.

Figure 2.1: Layered plywood track with keel channel.

To fit the track inside the plastic tubing, the bottom of the track was hand cut and honed in an effort to match the contour of the tube. Before placing the track inside the tubing, 90 ° aluminum strips were fitted inside the channel and secure to the top of the track as shown in Figure 2.2. This ensured a straight and smooth surface for the keel to rub against. Permanent magnetic strip tape was fitted on top of the aluminum strips and ran the length of the track. The magnetic strips were placed as close as possible to the inside of the channel but with enough clearance so the keel of the pod would not touch the strips when placed on the track. Placement of the magnetic strips on the track and pod came from the observations from the first preliminary track.

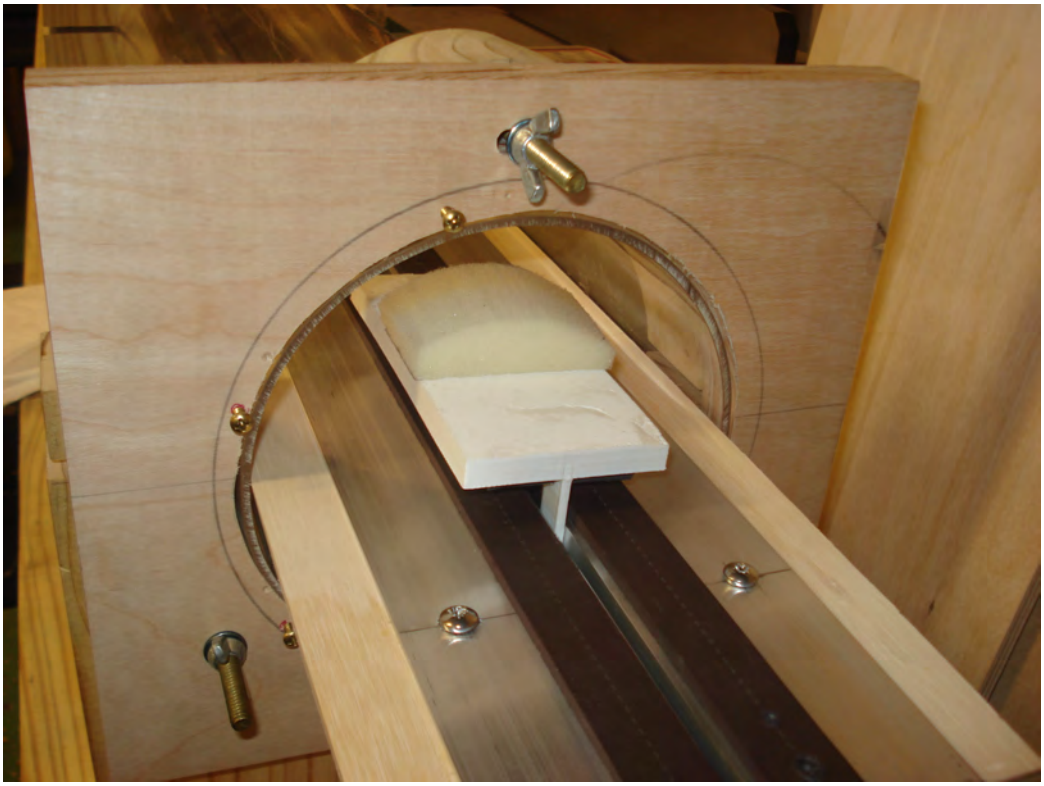


Figure 2.2: Final Track with magnetic strips and aluminum channel.

Plastic tubing used to encapsulate the track came in three sections. Once the track was placed inside the tubing, the sections of the tubing were secured together by hand made wooden clamps fashioned from leftover plywood material. The sections were clamped together and the track was secured to a 2"x10"x10' platform. A straight board was fitted to run under the track and tubing to strengthen the track and help prevent sagging. A channel was cut on the bottom side and runs the length of the strengthening board. This was used to hold wiring of the DC fans and the position sensor as shown in Figure 2.3. A square wooden face plate was mounted to each end and is used to secure the DC fans to the tunnel.

Figure 2.3: Final Track with re-enforcement underneath.

### 2.2.2 POD construction

The wooden pod seen in appendix A.0.1 was quickly replaced with a 3D printed plastic version. The plastic base is light-weight and generates less friction on the keel than the previous wooden version. Figure 2.4 below shows a drawing of the pod base that was created using Solid Edge software.

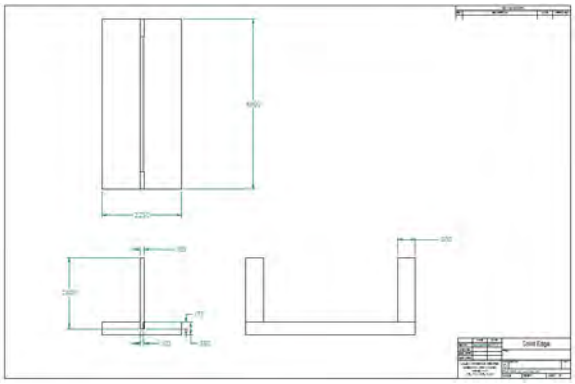


Figure 2.4: Solid Edge drawing of 3D printed pod base.

Evolution of the pod came from preliminary testing on the first and second tracks. It was noted during open loop testing with the DC fan that a light- weight pod with thinner keels showed the best response. Figure 2.5 exhibits the pods evolution.

Figure 2.5: Pod Evolution.

After the base of the pod was printed, foam was fitted on top of the pod to eliminate as much void as possible between the pod and the inside of the tunnel.The foam will act as a sail once inside the tube, giving the pod more surface area for the air pressure to act upon. Figure 2.6 below shows the finished pod.

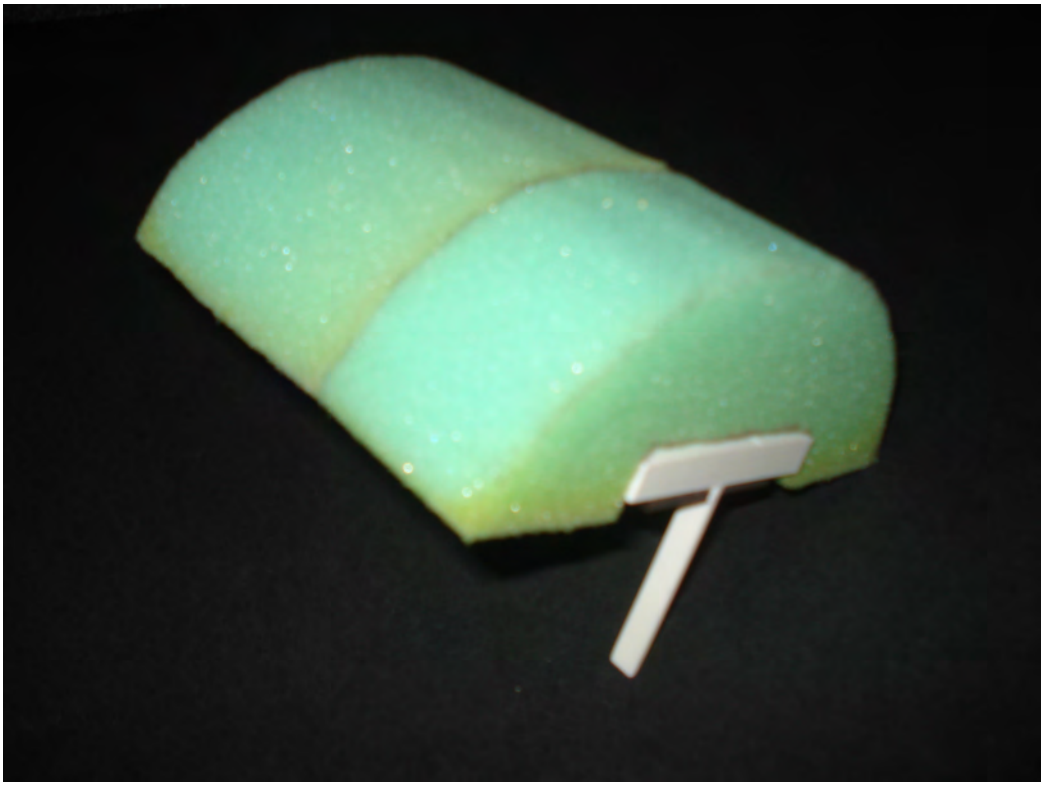


Figure 2.6: Solid Edge drawing of 3D printed pod base.

## 2.3 Electronics

Power and communication between electronic components was handled with off the shelf equipment and hand soldered boards. Figure 2.7 shows the electronic component used in this project.

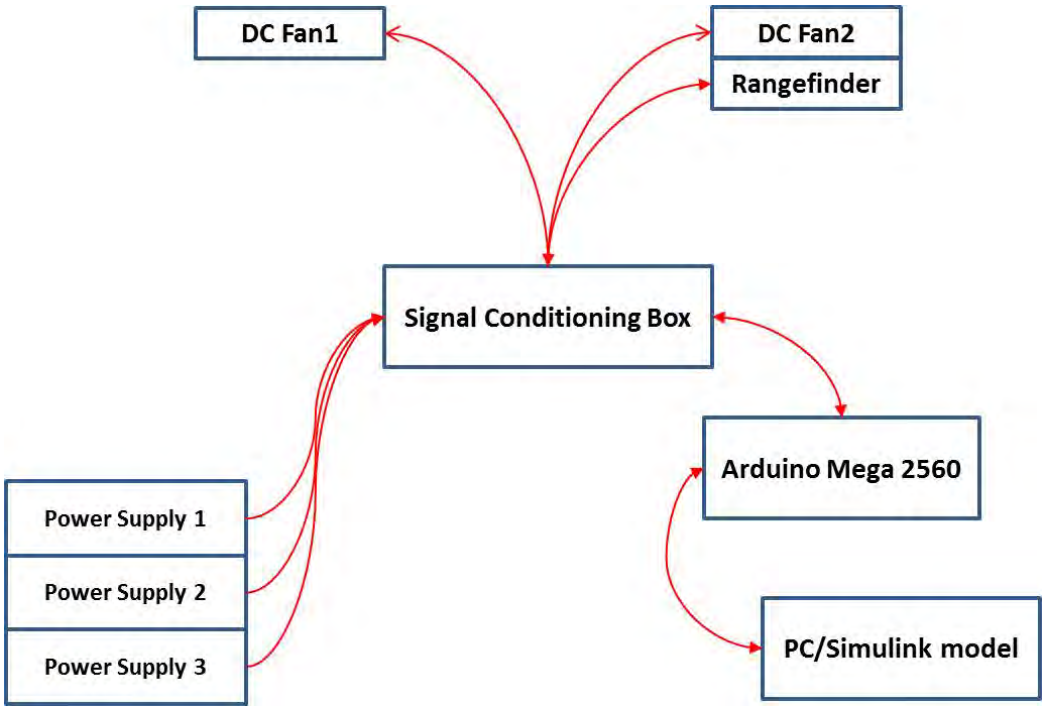



Figure 2.7: Electronic Flow Chart.

### 2.3.1 DC Fans

Once the pod is placed into the tunnel, airflow is the only force controlling the position of the pod. To provide this airflow, two DC fans were selected. The DC fans were selected based on the cubic feet per minute (CFM) output, DC voltage requirements and resource limitations. The airflow of these fans is listed as 235 CFM at 3900 rpm's and operate on 0-24VDC provided by external power supplies. A photo of the fan can be seen in Figure 2.8 . It is worthy to note at this point that the fans are polarity protected. This limits their use to pushing air in one direction only. A snapshot of the datasheet for the fans can be found in appendix B.0.1. Also worthy to note that the fans provide no return signal and thus a feedback loop on the fans will not be implemented.

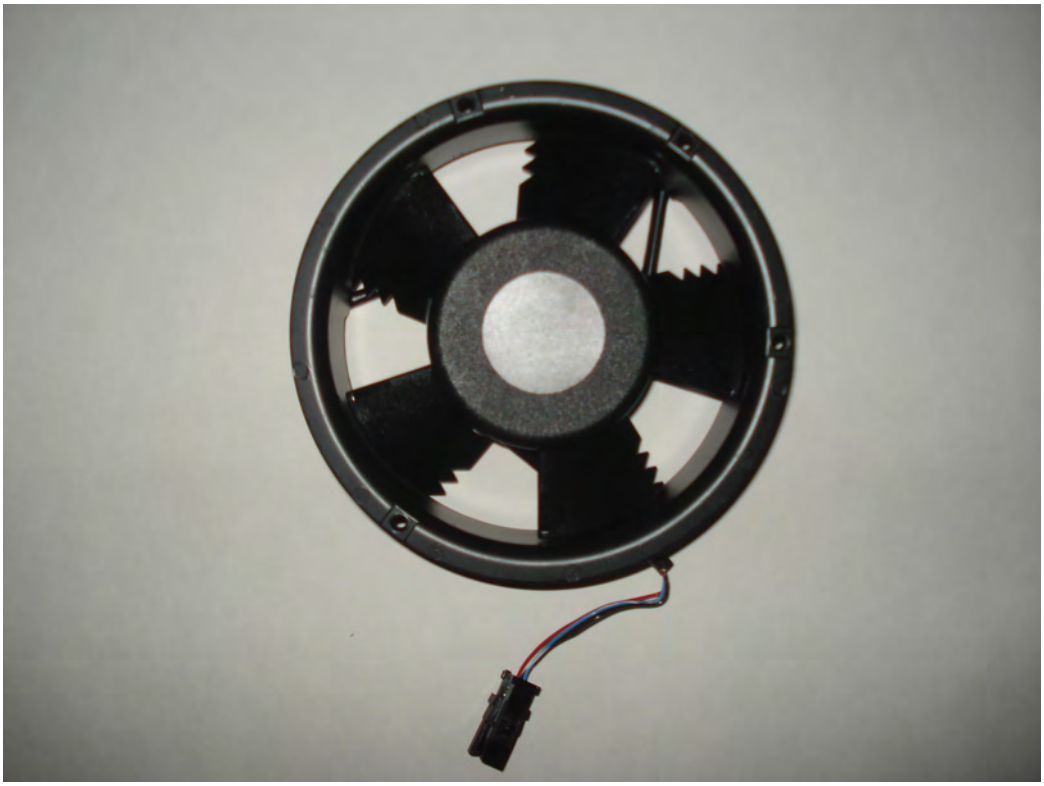


Figure 2.8: DC Fan.

### 2.3.2 Power Supply

Preliminary tests revealed that one power supply is not sufficient to meet the power needs while running the entire system. The current needs taken from the DC fans datasheet proved to be larger than anticipated when operating the fans sequentially. These larger than expected current swings disrupted the position sensor and temporarily disable accurate readings. For this reason, the decision was made to place each DC Fan on it's own power source as well as the position sensor. This provides power isolation for each component across the entire project.

The first power supply is hand built and is used to provide the 5 VDC necessary to operate the position sensor. A picuture of the handmade power supply is shown in Figure 2.9. For greater detail on the handmade power supply see appendix C.0.1 and C.0.2.

Figure 2.9: Handmade Power Supply.

Two, off the shelf DC power supplies were used to provide a constant 24VDC to drive the fans. These supplies are capable of producing 0-30VDC at 0-5amps. Details of the power supplies can be found in appendix C.0.3. During simulation, these power supplies provide constant power to the amplifiers located inside the signal conditioning box.


Figure 2.10: Off the Shelf DC Power Supplies.

### 2.3.3 Signal Conditioning

A signal conditioning box was built to house a number of handmade circuits that provide several functions and is shown in Figure 2.11. Banana jacks were used to pass signals and power in and out of the box.

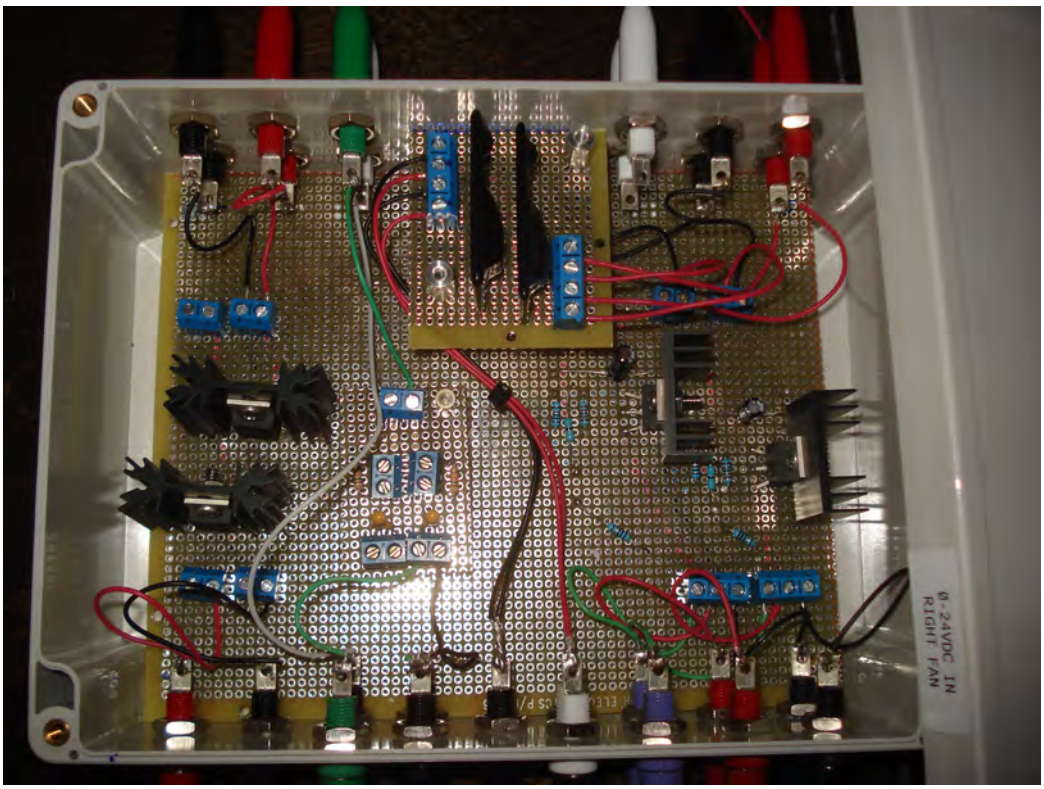



Figure 2.11: Signal Conditioning Box - Top view.

The first circuit featured on the far left inside the signal condition box was created to provide a consistent 5VDC to the position sensor (see appendix D.0.1). This circuit contains a 5V linear regulator and is wired in the same manor as suggested by the manufacturer's datasheet (see appendix D.0.2) and was built so that the DC voltage coming in from the power supply could be adjusted without having to recalibrate the position sensor. This circuit was built out of necessity, after having recalibrated the position sensor several times due to voltage adjustments.

The second circuit in the lower middle is a simple R/C low-pass filter circuit (see appendix D.0.3). For a diagram view of the circuit see appendix D.0.4. This series wired circuit uses a 15K ohm resistor and a 1 microfarad capacitor. The resistor/capacitor circuit provides a 10Hz filter for the returning analog signal of the position sensor. This design was determined based on the position sensor datasheet and the need to gaurd against signal attenuation. The use of terminal blocks were used in this circuit to hold the resistors. This allows the user to easily insert and remove different sized resistors during preiminary signal testing and helped produce the best results. The signals are passed through banana jacks and returned to the Arduino boards ground and analog input.

The third circuit consist of two solid state relays (see appendix D.0.5). This need was realized after a series of preliminary test concerning the DC Fans. When one fan pushed air into the tunnel, the opposite turned in reverse. This reverse direction caused the DC motor of the opposite fan to produce a voltage which bled back to the signal amplifier. To isolate this back voltage, power from the amplifier must first pass through the solid state relay. Control over the relays is handled through MATLAB/Simulink software that triggers a PWM signal from the Arduino board. When voltage is commanded to the fans, a PWM signal is simutaneously sent to the solid state relays, the circuit is closed and current flow to the fans is permitted. For a wiring diagram see appendix D.0.6.

The forth and final circuit in the signal conditioning box contains two power amplifiers (see appendix D.0.7). Continuous 24VDC power is provided by external power supplies to the power amplifiers. The amplifiers control a range of 0-24VDC to the fans by means of a PWM signal sent from the Simulink software to the Arduino board and out to the amplifiers. Based on the 0-5V PWM scaling, the amplifiers then permit a voltage range of 0-24VDC to be passed through to the DC Fans. The power amplfiers are wired according to the manufacturer's datasheet (see appendix D.0.8).

### 2.3.4 Position Sensor - Rangefinder

In order for the position of the pod in this thesis to be controlled, the system must first realize where the pod is. This was accomplished through a laser rangefinder shown in Figure 2.12 .

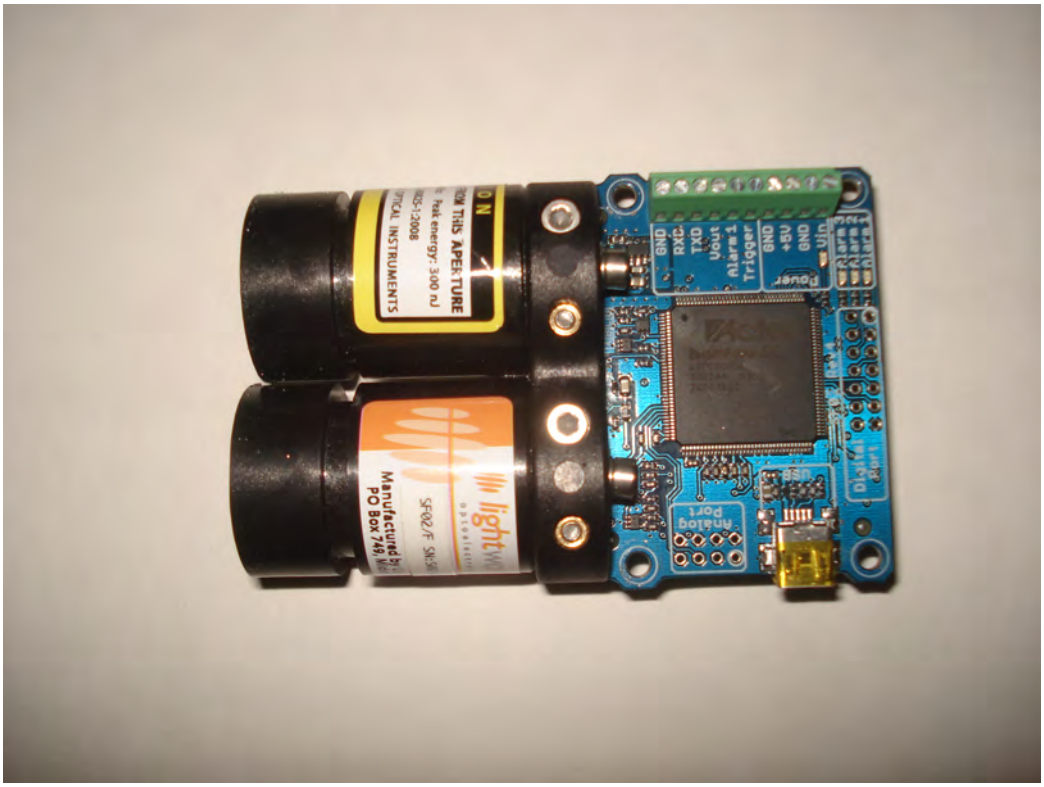



Figure 2.12: Rangefinder - Top view.

The rangefinder has an array of settings for different types of setups and can be made through software provided by the rangefinder manufacturer. One of these settings gives the rangefinder the ability to run in both a filtered mode and an unfiltered mode. The unfiltered mode measures distances up to 50 meters at 32Hz but periodically has large, out of range readings. The filtered mode can measure distance up to 50 meters at 12Hz but has less out of range readings. Another setting gives the rangefinder the ability to set a max distance to read. The setup screen for the software can be seen in Figure 2.13.

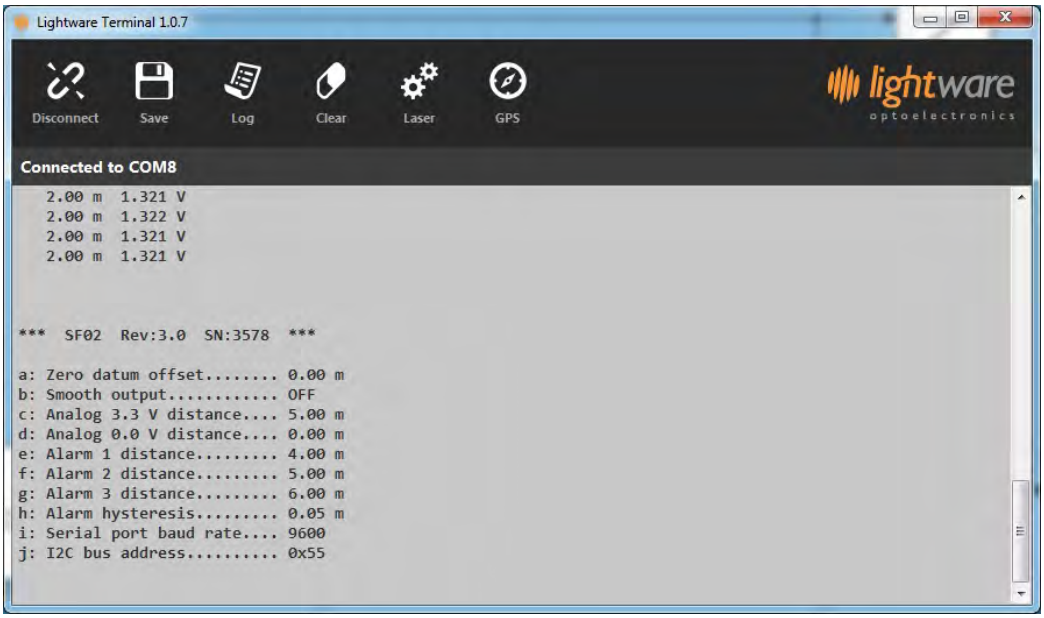



Figure 2.13: Rangefinder - Setup Display.

For this thesis, the rangefinder is setup in an unfiltered mode to measure a distance of up to 5 meters. The decsion to set the distance measure to 5 meters allowed scaling of the voltage to be greater between each reading and thus give greater distinction between measurements.

Running the rangefinder in an unfiltered mode was based on the characteristic behavior of the system. The system in this thesis has a low reaction time. In short, the fans can produce air pressure but are unable to reduce the pressure inside the tunnel necessary for fast correction swings. It was reasoned as acceptable to have peroidic, large swings from the rangefinder due to the fact of slow system response.

### Rangefinder calibration

Calibration of the rangefinder was made using a long 2"X4"X10' wooden board. Small $\frac{1}{4}$" cuts were made cross-wise at each inch for a total of 110 cuts. The cuts were made to hold a small vertical panel that will be used by the rangefinder to record distances. Figure 2.14 shows a picture of the calibration setup.

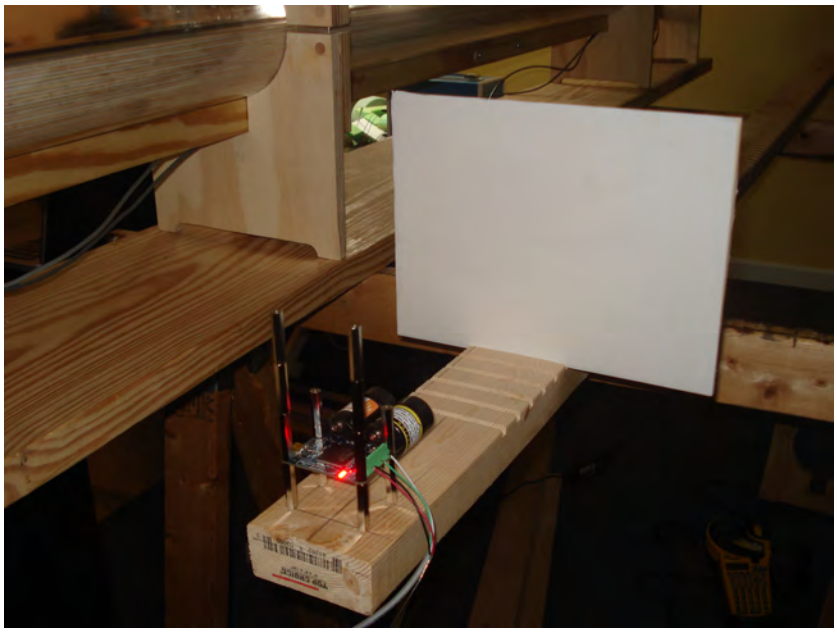


Figure 2.14: Rangefinder - Calibration setup.

With 5VDC supplied to the rangefinder, calibration began by first creating a simple Simulink model which would record the various distances measured by the rangefinder. The Simulink model is shown in Figure 2.15 and features the custom package Simulink/Arduino analog block output to a display.

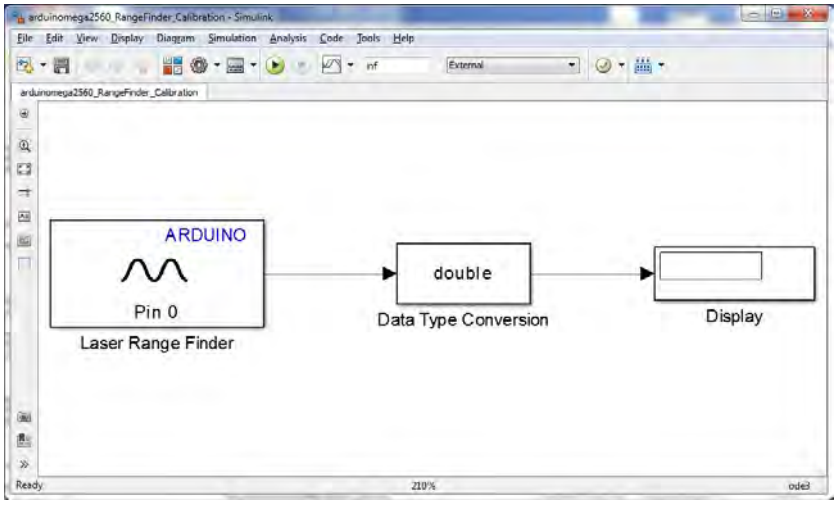



Figure 2.15: Simulink - Calibration Model.

The calibration model was run and the distances recorded at each inch. A MATLAB script was written to run the model and record the distance which were saved to an EXCEL sheet. The script was setup to record (5) measurements at each inch (100 inches total) with a 5 second delay between each recording. This allowed outlying ranges to be identified during post-processing and provided repeatability of calibration. Code for the calibration is in appendix E.0.1

After the calibration script was run, post-processing of the measurements in EXCEL were performed. An EXCEL scatter plot provided a linear fit of the data points. Figure 2.16 shows the recorded analog measurements on the x-axis and the inch mark for that measurement on the y-axis. The linear fit equation is used later in the Simulink model to report position.

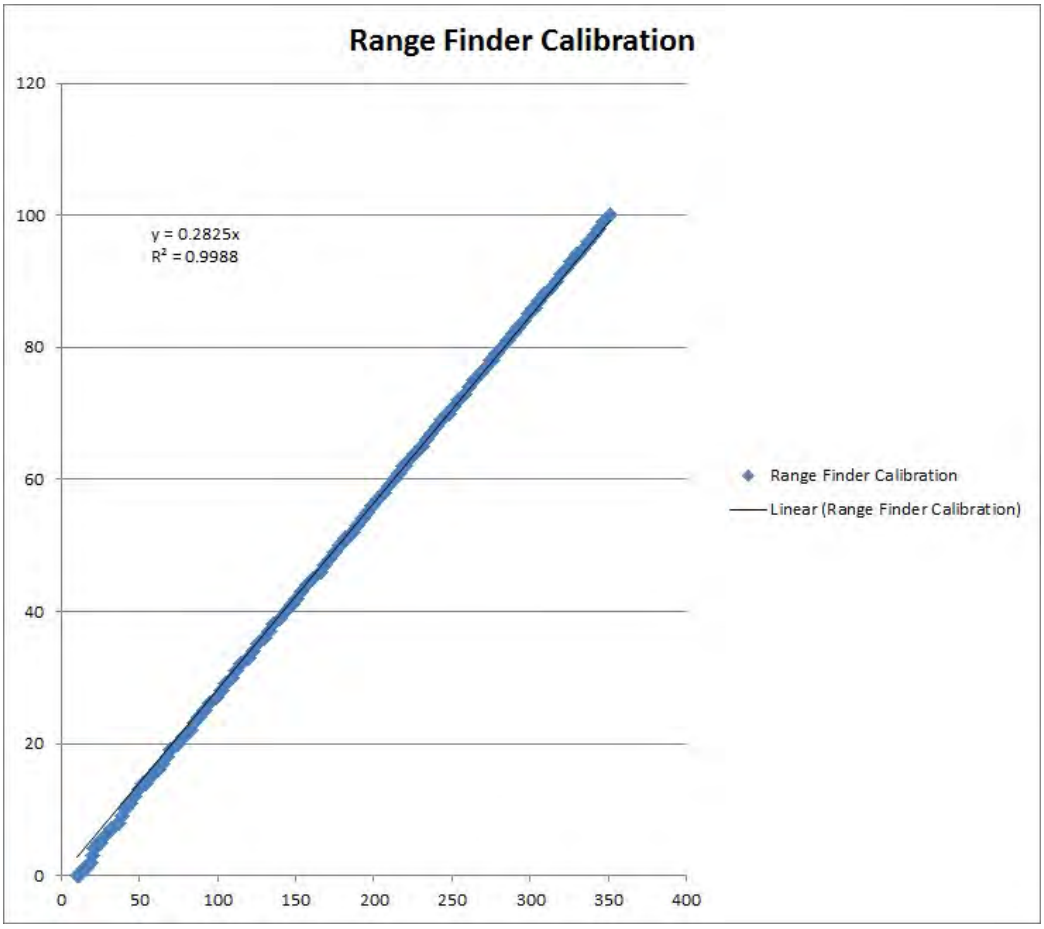



Figure 2.16: Recorded Data with Linear Fit Equation.

### 2.3.5 Arduino Mega 2560

In this thesis, an Arduino Mega was used. The Mega board contains an ATmega1280 microcontroller and is equiped with 54 digital input/output pins, 16 analog pins and 4 hardware serial ports. For this thesis, the Mega is used for hardware communication, data acquisition, and simulation. A full description on this board and processor can be found from the Arduino webpage. [2].

The layout of the board for this project can be seen in Figure 2.17. Four PWM pins are used, two to control the left fan and two to control the right fan. Pins D7 and D5 are used from the board to send the correct scaling to the amplifiers which in turn control the voltage sent to the fans. Pins D3 and D4 are sent simutaneously out to the solid-state relays that close the connections to the fans and allow current to flow.

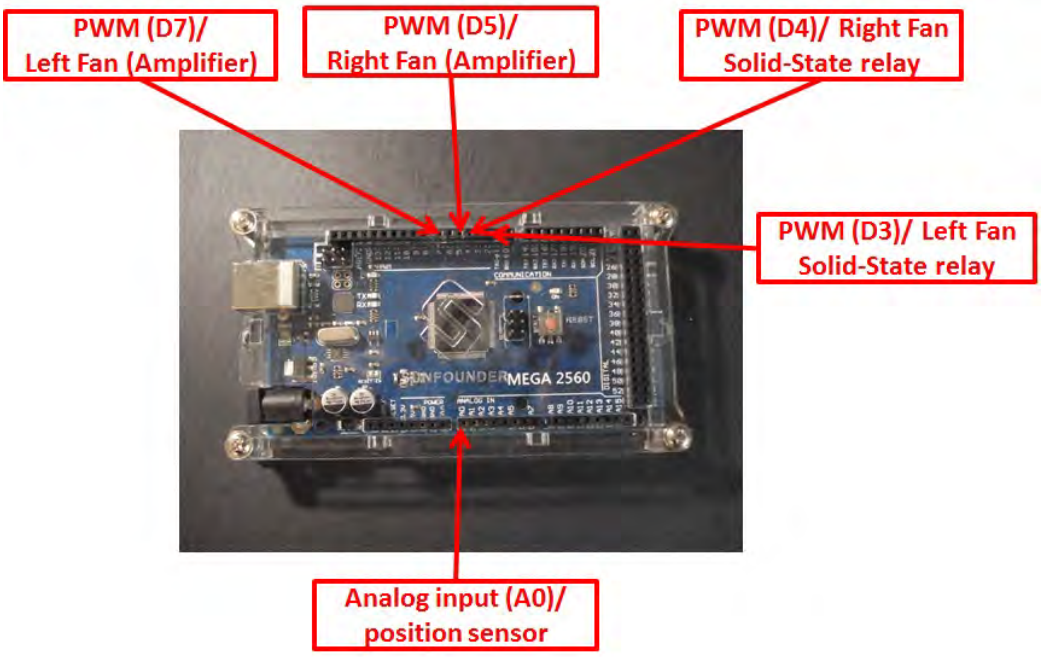



Figure 2.17: Arduino Board Layout.

Analog pin (A0) is an input pin and receives a millivolt (mV) signal from the rangefinder. This signal is sent back to the Simulink model, through the Arduino and is used to report the pod position.

## 2.4 Software

MATLAB/Simulink software was used to create the model of the system and develop a control loop. The MATLAB/Simulink software is a product created and licensed through the company Mathworks. MATLAB software is used throughout the world by Scientist, Engineers and students for modeling and simulation[4]. The Simulink development environment provides users with easy to use toolbox packages that allow fast and robust prototyping of mathematical models. Their software environment is used across a large number of industries and can be applied for purposes such as: algorithm development, mathematical computation, data analysis, signal processing, engineering graphics and application development to name a few.

MATLAB/Simulink was selected for use in this thesis for two reasons. First Mathworks has provided an add-on software package that allows users to communicate with the Arduino Mega that is also used in this system. The software can be downloaded directly from the MATLAB application and the Simulink toolboxs appear directly in the software "user library". The toolbox provided blocks that allow direct communicate between the Simunlink model and the Arduino Mega.

Secondly, MATLAB can be used in a cradle-to-grave manor for this thesis. Communication with the Arduino processor, modeling, development and simulation of the system, and post-process analysis can be performed through one software outlet.

### 2.4.1 MATLAB/Simulink Model

There are two main MATLAB/Simulink models for this thesis. The first is a low fidelity, open loop signal/system analysis model shown in Figure 2.18. Several experimental test (covered in section 4.1) are performed using this model. See appendix F.0.2 for a closer look and breakdown of functionality for this model. This model provides a look at system responses and helped to formulate a better understanding of how the system would work as a whole before closed loop modeling began.

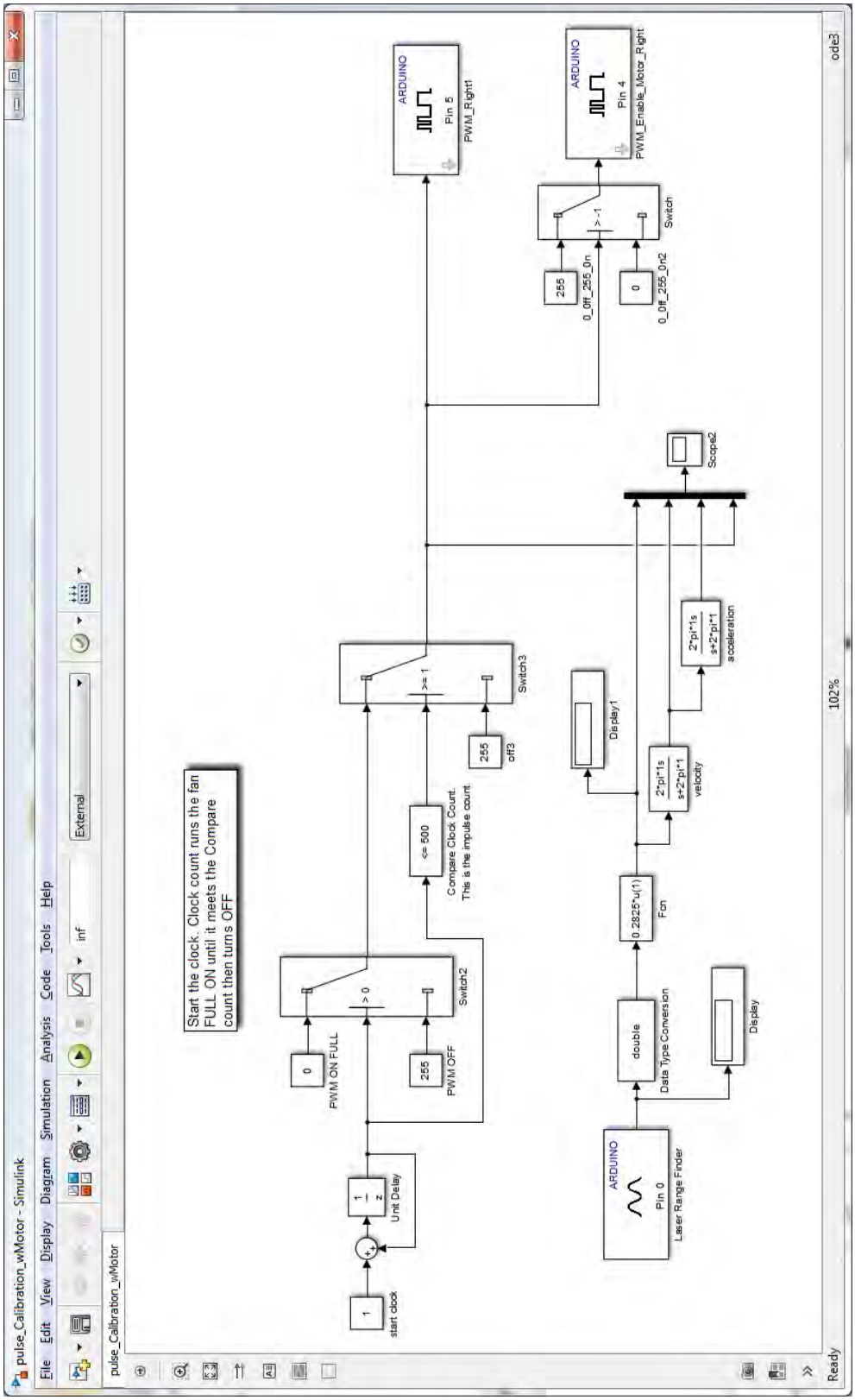



Figure 2.18: Open Loop - Signal Model.

The second main model is the closed loop and final model see Figure 2.19. The closed loop model will be used for final results testing and is explained in full in appendix **??**.

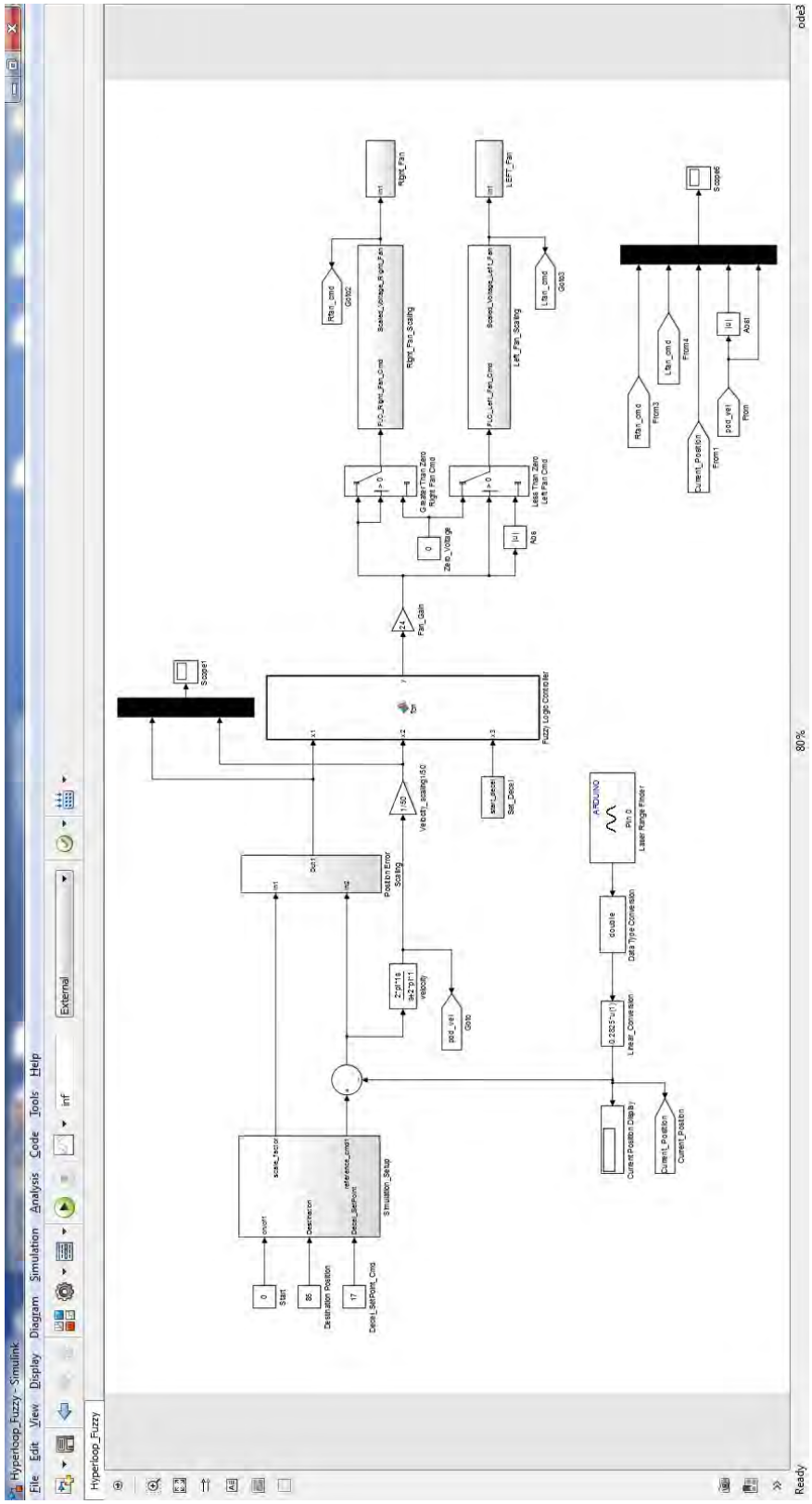



Figure 2.19: Closed Loop - Hyperloop Model.

# Fuzzy Logic Control

## 3.1 Fuzzy Logic Introduction

The words “Fuzzy Logic Control System” are often met with the feelings of doubt and hesitation.“Fuzzy” and “Logic” in the same sentence does not seem to sit well with Engineers. However, Fuzzy Logic control theory is changing the way modern Control Engineers think about control systems. One doesn’t have to look too far to realize the world is not a linear place. Many control applications involve the breaking down of complex, non-linear physical problems into manageable linear equations which describe a system mathematically. The mathematical equations are then integrated together to form a complete system. The problem with this scenario is that the more complex those systems are, the ability to control those systems in a linear way becomes increasingly difficult if not impossible. This was noted by Dr. Lofti Zadeh.

In 1965, Dr. Lofti Zadeh[10] introduced an idea to deal with complex systems by implementing human reasoning to describe a system. In a 1973 paper titled "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes" [10], Dr. Zadeh stated that, "As the complexity of a system increases, our ability to make precise and significant statements about its behavior diminishes until a threshold is reached beyond which precision and significance become almost mutually exclusive characteristics".

Exact percision cannot always be achieved in the natural world. The answers are not always “yes or no”; “hot or cold”; “black or white”. In many cases there are shadowy

gray areas that may often be missed simply because the lack of tools or insight to describe something that is not on a palette with only black and white colors. In each case, there are degrees of understanding. The term to describe the temperature of something as “hot” may vary from person to person. Extremes are usually not hard to define. Most people would define boiling water as “very hot” and water just above freezing as “very cold”. However, if the water temperature between is varied in those extreme points, one would get a multitude of answers when asked to describe the water. Mathematically the temperature of the water can be measured to come up with a hard number but that would not describe the water when asked if it was warm. In this case, the water temperature between the extremes would become “fuzzy” when trying to describe the water mathematically. This is a perfect case when fuzzy logic can be considered.

In this example, extreme water temperature can be described Linguistically as very hot and very cold. In the mathematical world, boiling water is described as 100 degrees celsius and very cold water as 0 degrees celsius. In fuzzy logic, the use of linguistics to describe aspects of the system will allow the user to later characterize the areas between the extremes. In Figure 3.1 the use of linguistic variables is used to describe the temperature of water.

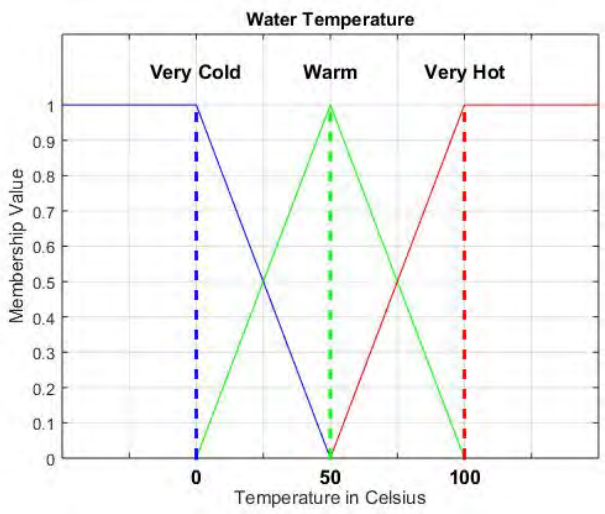



Figure 3.1: Linguistic example with partitioning.

In Figure 3.1 the membership value shown on the left side of the graph or y-axis range from 0 to 1. Mathematically, the extremes of very hot or very cold are shown on the x-axis, displayed in celsuis and range from 0 to 10 with warm water being exactly halfway in between. Vertical dashed lines are inserted to show the "fuzzy partition" between the 3 linguistic temperatures. By partitioning it is shown that everything to the left of the dashed blue line can be considered "very cold" and everything to the right of the dashed red line can be considered "very hot". As the temperature changes from one partitioned subspace to the next, the transition is smooth or rather as the temperature increases from very cold to warm it is a gradual effect. Mathematically, if the temperature were to be measured at 75 degrees celsius the membership vector would be [ 0, 0.5, 0.5]. This shows that the temperature is NOT very cold but that is has a membership value of 0.5 for warm and 0.5 for very hot and the sum of the membership is equal to 1. In this example, the temperature of the water is now broken down into three fuzzy partitions, "very cold", "warm", or "very hot". In conditions of overlap, it is assumed that a maximum of only two partitions or two membership functions can ever be active at any one instance.

So how is a Fuzzy Logic Controller built? Fuzzy logic is comprised of four basic elements; fuzzification , inference, fuzzy rule base, and defuzzification. Scaling both the input and output of a Fuzzy Logic Control System (FLC) allows the user to work with values between [0, 1] as can been seen in Figure 3.1. To understand this process the blocks of the Fuzzy Logic Controller are shown in block diagram form in Figure 3.2.

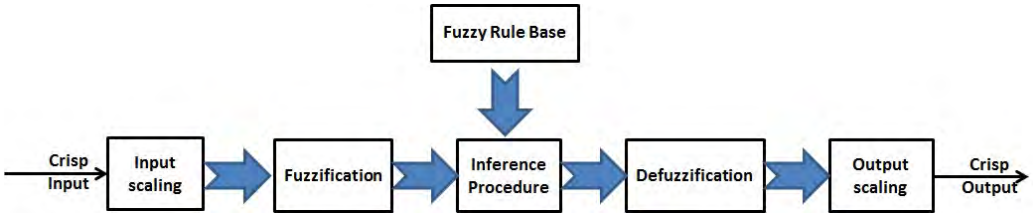



Figure 3.2: Block diagram of Fuzzy Logic Controller.

- Scaling - provides an easy way to deal with inputs/output on the basis of a 0 to 1 scale. Scaling requires some prior knowledge of the systems input and output capability. Scaling is not necessary but is often used for simplicity.

- Fuzzification - The process of taking a crisp input value and determining which fuzzy set the input value lies within. In the above example of the water temperature; if the input value for the temperature is 75 degrees celsius, then that input value is part of the fuzzy set of “warm” and “very hot”. The membership vector would be [0, 0.5, 0.5]. If the input temperature were to be 60 degrees celsius, the membership vector would be [0, 0.8, 0.2]. Figure 3.3 illustrates this membership.

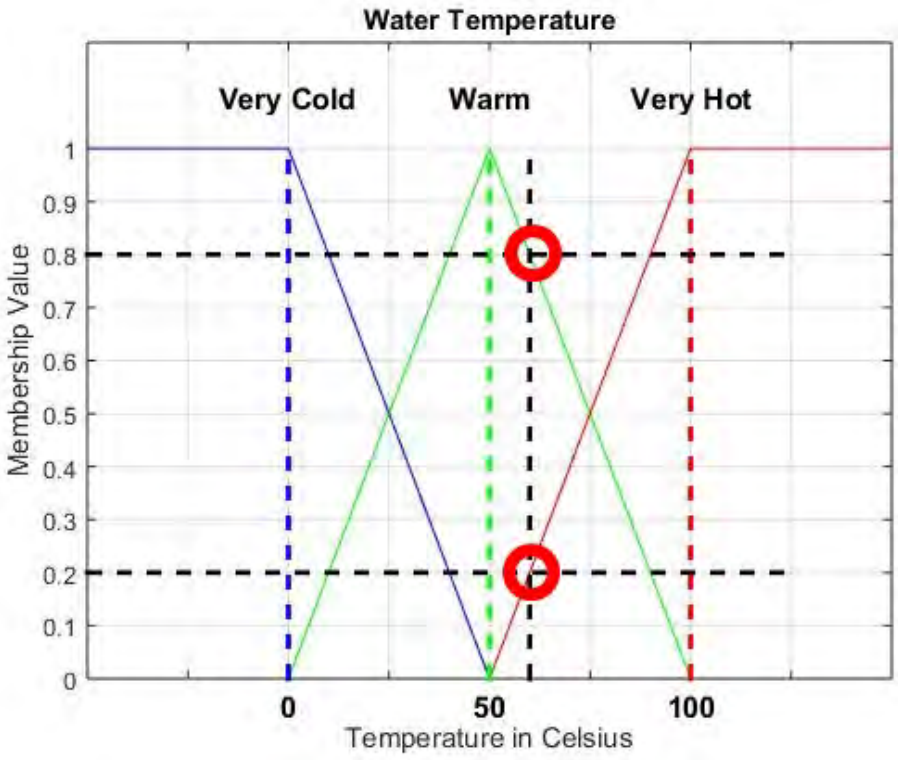



Figure 3.3: Fuzzification Process.

In Figure 3.3, it is easy to see that the temperature of 60 degrees celsius is more warm than very hot with 0.8 membership on the warm side and only 0.2 membership on the very hot side. Again, the sum of the membership values is 1. The equations used

to determine the membership are seen in Figure 3.4 [24]

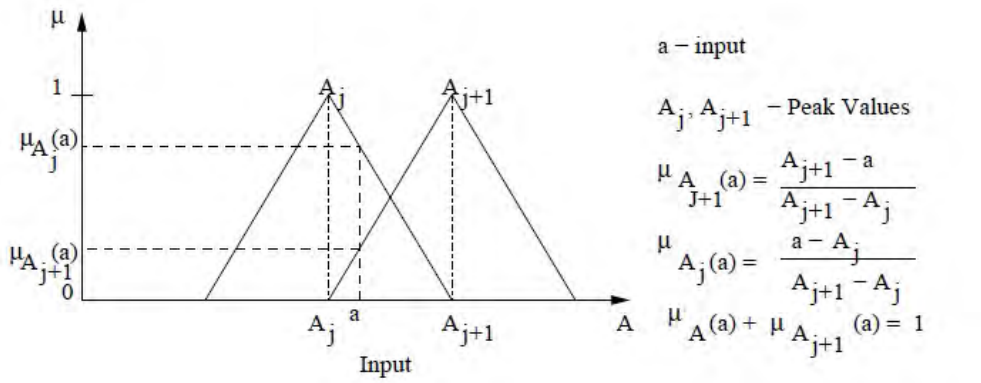



Figure 3.4: Fuzzification Equations.

- Fuzzy Rule Base - The fuzzy rule base is simply a set of (if-then) rules. Based on the input, an output value is passed based on the rule base. For instance, (in the water temperature example) if the input temperature is determined to be too high based on the rule base, an output command may be sent that would lower the water temperature. For example, a set of control rules to control the water temperature as warm might be written using if-then rules as:

  *if temperature is very cold, then send a command to raise the temperature*
  *if temperature is warm, then send a command to leave the temperature the same*
  *if temperature is very hot, then send a command to lower the temperature*

- Inference Procedure - The inference procedure is the decision making process by which it is determined the truth value for each rule and applies this value to the "then" statement of the rule base. There are several ways to determine this but the three most commonly used are the product-sum method, the product-max method and the min-max method. For detail discussion of these methods refer to [25].

- Defuzzification - The final step in developing a fuzzy logic controller is to match a scaled output based on the degree of match by the inference process. For instance in the water temperature example, if the desired temperature is 50 degrees celsius and the input is 60 degrees celsius. From the fuzzification section, the input is closest to the "warm" membership than any other membership. A command to lower the temperature should be the output based on the rule base. The inference process has determined the strength by which the temperature rate is changed based on the partitioned membership. A crisp output is commanded from the output fuzzy set, based on the strength determined by the inference process.

## 3.2 Fuzzy Logic Implementation for Hyperloop

Three fuzzy sets are used for control of the hyperloop and are implemented in chapter 4, section 4.3. The first two sets are used for input values, and the third fuzzy set is the output fuzzy set. The first input fuzzy set is the position error. Because the position sensor is located in a fixed position, the error comes in both positive and negative values. The second input fuzzy set is the rate of change in the position error or the derivative of the position error. The output fuzzy set determines the voltage command to the fans, which in turn controls the air velocity running into the tunnel and acting on the Pod. The linguistic sets are shown below in Figures 3.5, 3.6 and 3.7. The linguistic fuzzy sets are defined as; "NB" for negative big, "NM" for negative medium, "NS" for negative small, "Z" for Zero, "PS" for positive small, "PM" for positive medium, and "PB" for positive big.

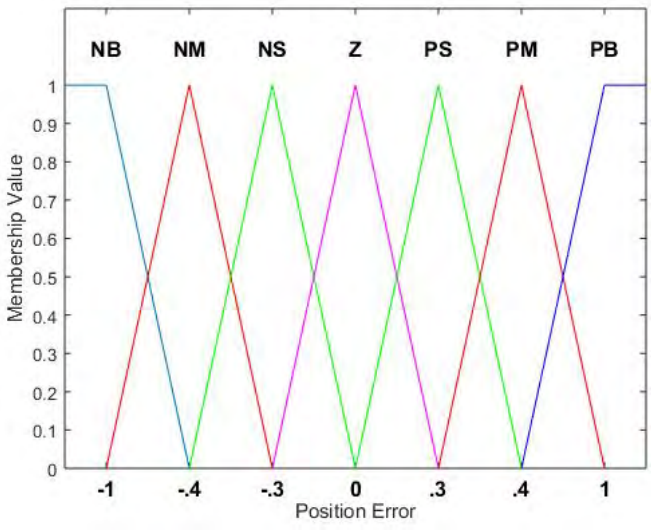



Figure 3.5: Position error fuzzy set.

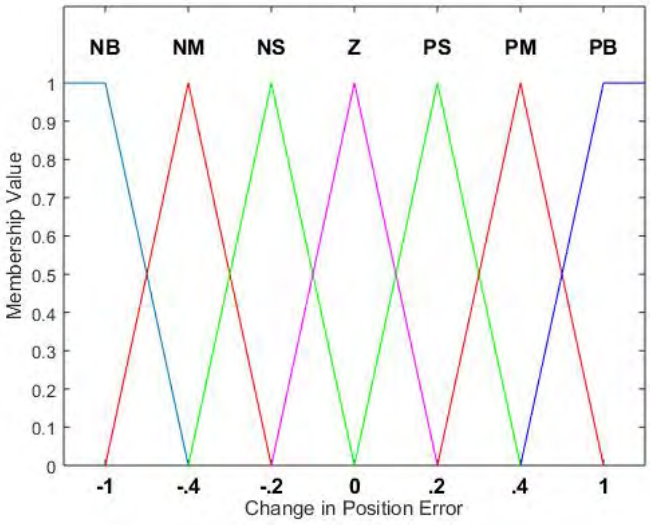



Figure 3.6: Change in position error fuzzy set.

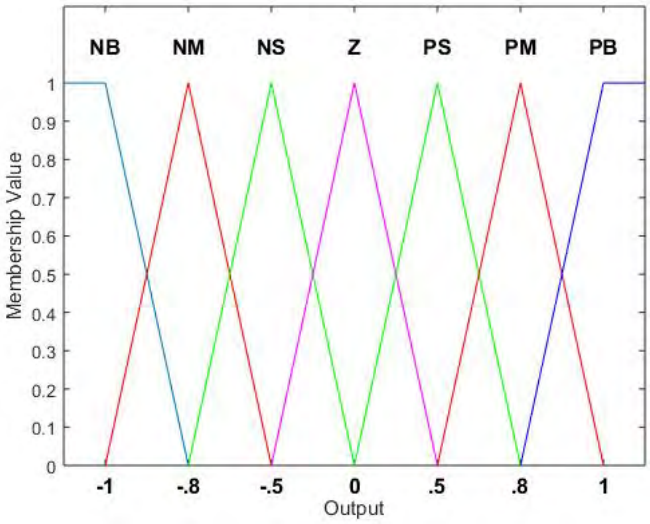



Figure 3.7: Output fuzzy set

If-then rule base describes the relationsip between the input fuzzy sets and the output fuzzy set. For example, one of the 49 rules can be written linguistically as:

*if the position error is NB, and the change in position error is NB, the output is NB*

The 49 rules (7X7) can be conveniently written in the matrix form as shown in Figure 3.8. Looking at the rule base, the output is divided into two groups; a postive and a negative with zero output divided through the middle. The controller drives the positve outputs from the bottom right corner toward the center and the negative from the top left corner toward the center. Within the Simulink model, the logic is setup to control the right fan from positive values produced by the controller and the left fan is controlled by the negative values produced by the controller. However, this is merely due to naming and the current configuration setup of the physical model.

Fan Speed = Change in Position error \ position error

| Change in Position error \ position error | NB | NM | NS | Z | PS | PM | PB |
|---|---|---|---|---|---|---|---|
| NB | NB | NB | NB | NB | NB | NB | NB |
| NM | NB | NB | NM | NM | NS | Z | PB |
| NS | NB | NM | NM | NS | Z | PS | PB |
| Z | NB | NM | NS | Z | PS | PM | PB |
| PS | NB | NS | Z | PS | PM | PB | PB |
| PM | NS | Z | PS | PM | PB | PB | PB |
| PB | PB | PB | PB | PB | PB | PB | PB |

Figure 3.8: Linguistic Output Matrix.

# Experimental Method and Results

## 4.1 Step Response of the Open-Loop System

Testing began with a low fidelity model shown in Figure 4.1. Testing of signal I/O and pod reaction within the tunnel were performed and observed. The pod was placed stationary in the tunnel nearest the position sensor with no fan running and the signals were recorded for 5 seconds. Position, velocity, acceleration, and the commanded PWM signals are plotted using MATLAB code (Appendix F.0.5) and are shown in Figure 4.2. Note that the PWM signal on this plot is scaled 1:10 for viewing purposes.

Figure 4.2 shows the velocity swing in blue and even larger fluctuation in the acceleration in green. A closer look at the recorded position signal in Figure 4.3 clarifies these noted measurements.

From the position signal only plot (Figure 4.3) it can be seen that the signal fluctuates between 4.2" - 5.4" with the main concentration between 4.5"-5.1". The physically recorded distance was 4.8 inches. The fluctuation in the position sensor was deemed acceptable. However, using this signals derivative to determine the velocity and acceleration increased the amplitude swings observed in both. A closer look at the signals can be seen in Figure 4.4.

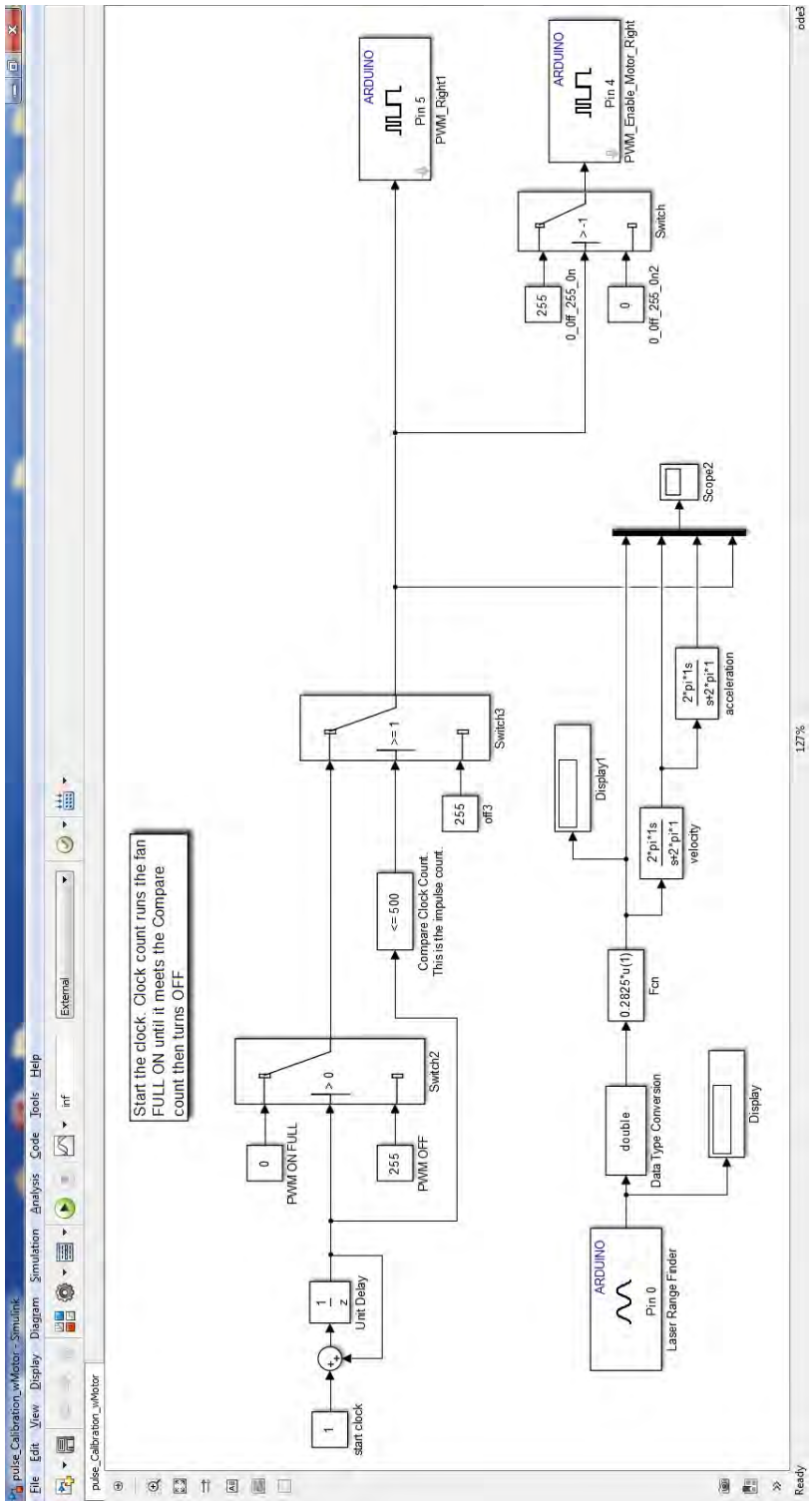



Figure 4.1: Low Fidelity Open-Loop Signal Model.

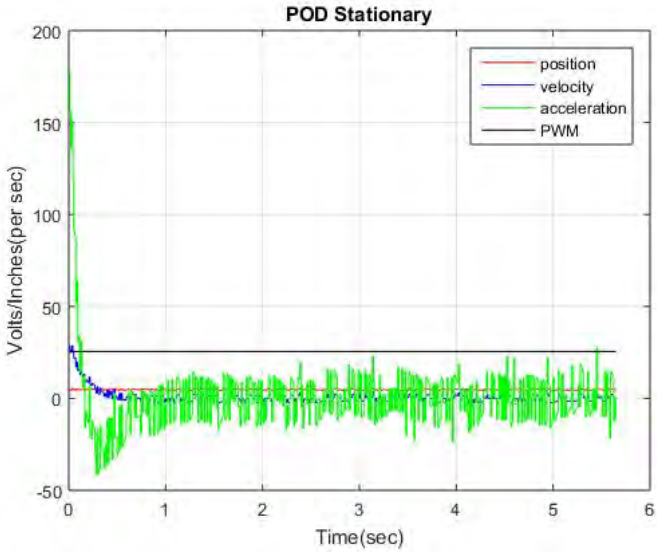



Figure 4.2: Low Fidelity Signal Mode - Stationary Pod Signals.

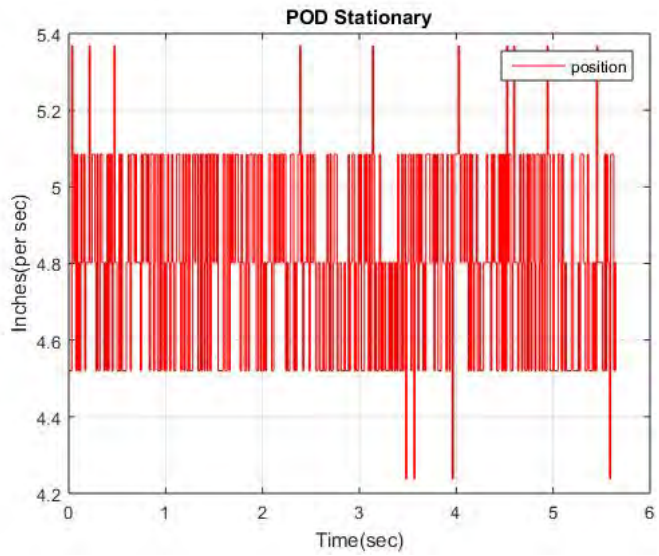



Figure 4.3: Low Fidelity Signal Mode - Position Signal Only.

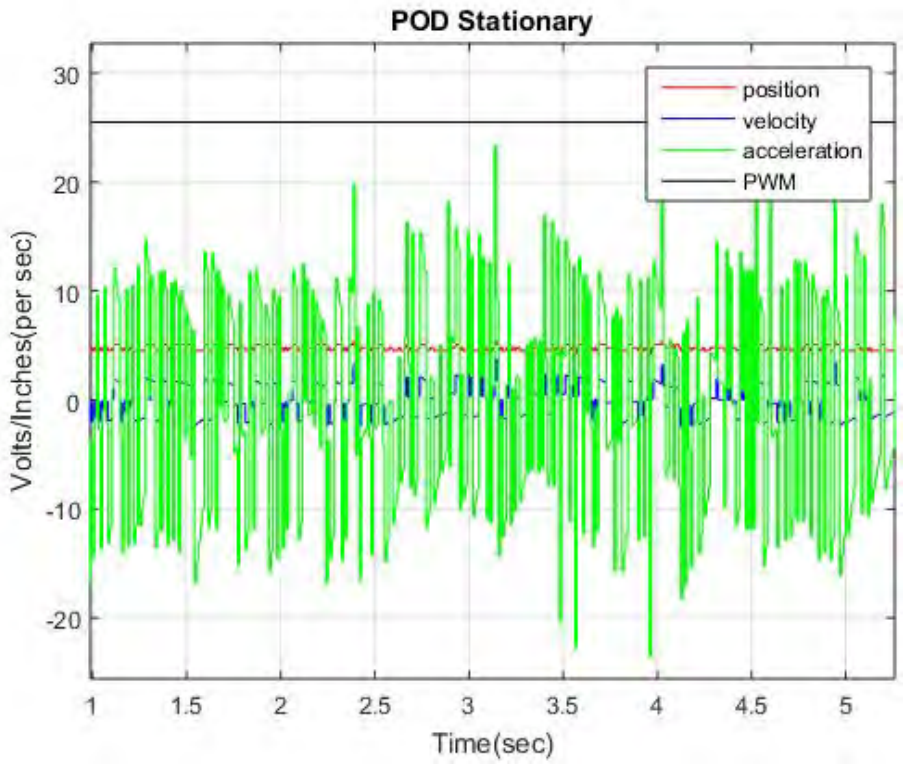



Figure 4.4: Low Fidelity Signal Mode - Signal Fluctuation.

A step command was given to the right fan and the position, velocity, acceleration, and the PWM signals were recorded in Figure 4.5. In this preliminary test, only the right fan was given a voltage command. A 24VDC command was sent to the right fan, the pod traveled the length of the track and was caught near the end.

A closer look at the step response reveals a delay. Figure 4.6 shows the voltage command come on at 1.8 seconds. Movement from the pod is not seen until 2.1 seconds into the simulation. This marks a notable consideration during test setup.

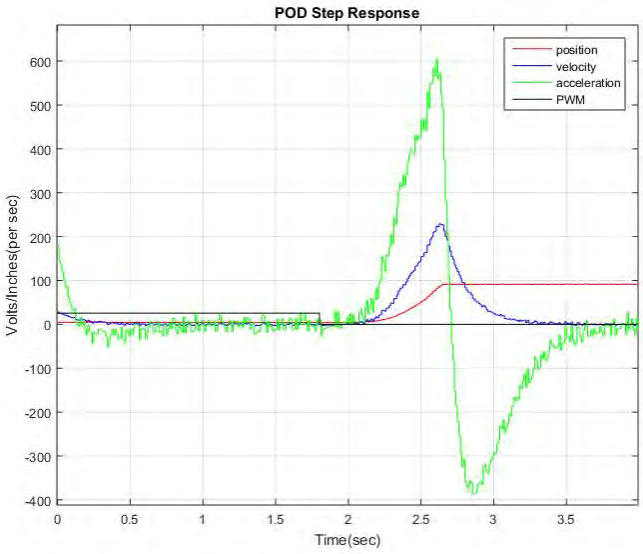



Figure 4.5: Step Response of the POD.

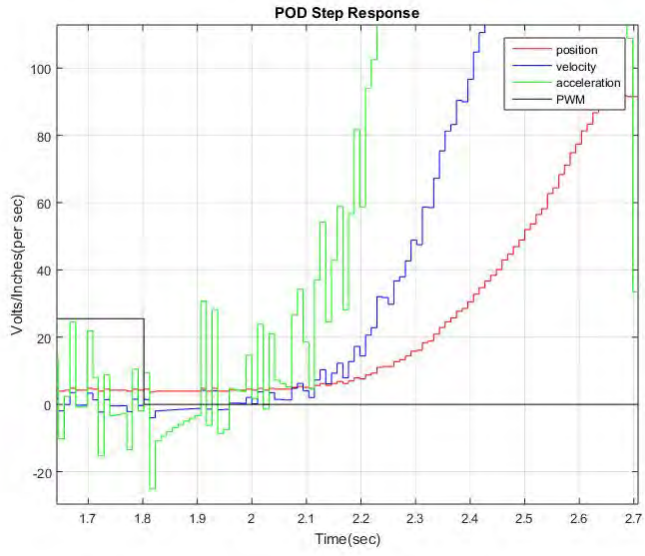



Figure 4.6: Step Response of the POD - Recorded Signal Closeup.

## 4.2 Experimental Method

One of the destinct behaviors noted was the reaction of the pod with both fans active. When one fan "pushed" the pod down the track and the opposite fan came on, there was a larger than expected response time from the pod. The opposite fan did not have an immediate impact on the pod and large current spikes were witnessed on the external power supplies. Due to the lack of resources and instrumentation, these spikes could only briefly be viewed and not measured. The current spikes on each fan ramped in excess of 2 amps. This prompted a hard system testing descion to be made. Only one fan would be permitted to operate at a time. With the lack of pressure venting in the system, the fans would "fight one another" to gain control over the pod.

Another behavior witnessed was the loss of recorded sample data. The Arduino Mega does not store data, so in order to record data for analysis, the Hyperloop simulink model needed to be ran in external mode. This is considered a debugging feature for simulation models. As noted from Mathworks website [22], using this feature increases the burden on the processor and could lead to tasks overrun.

## 4.3 Closed Loop Hyperloop Testing

The test below were performed using the closed loop model illustrated in Figure 4.7. As stated earlier, it is assumed that the Hyerloop pod would be traveling at a desired velocity before the Fuzzy Controller becomes active.

MATLAB scripts were used to run the model for each test. The scripts ensure model setup before simulation, repeatability, and organization of the data. These scripts can be viewed in sections: G.0.3 , G.0.4.

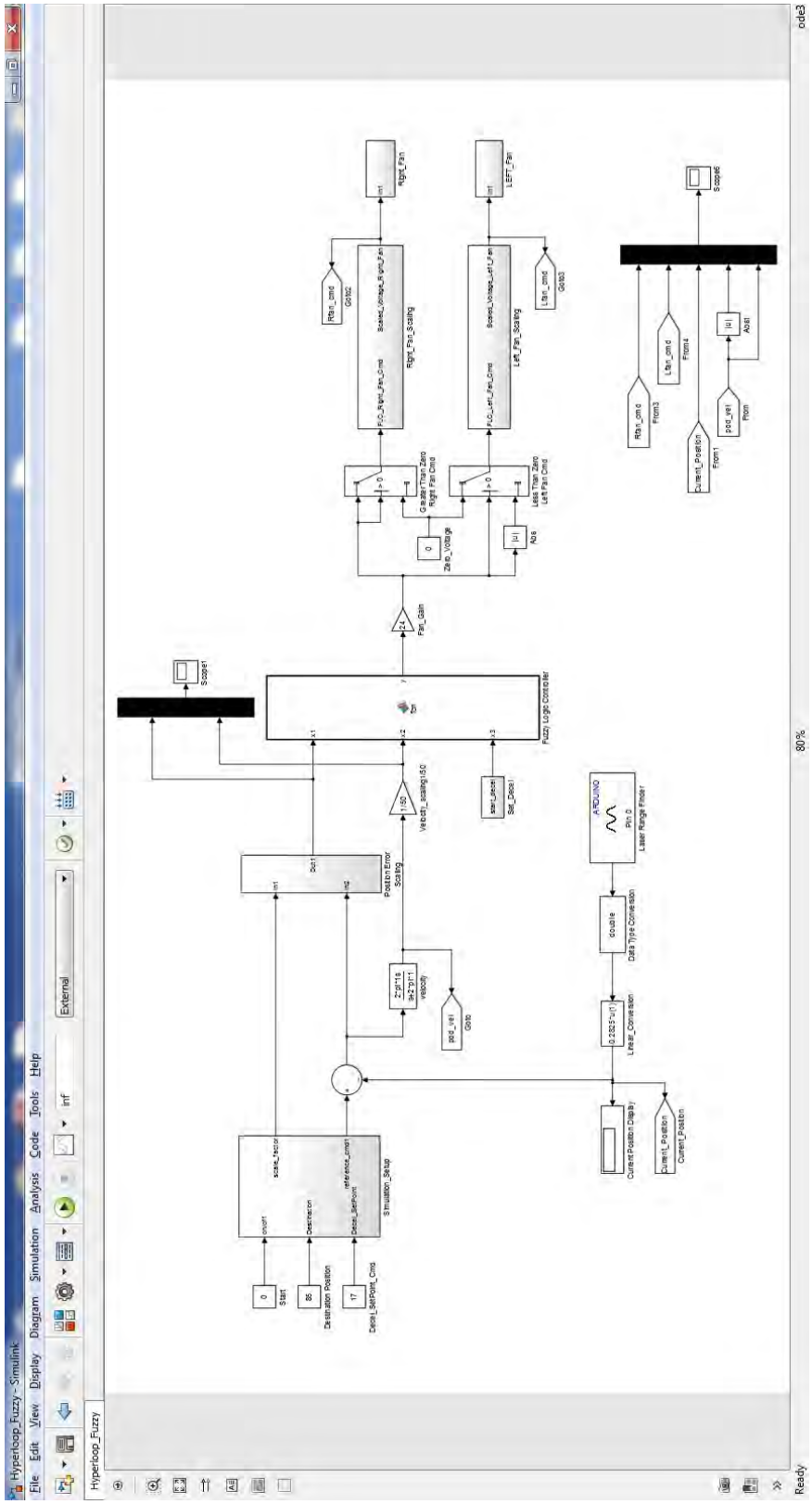



Figure 4.7: Closed Loop - Hyperloop Model.

The closed loop model was used to perform all final testing of the fuzzy logic position controller. The model can be broken down into four basic modules; reference command (4.3.1), feedback (4.3.2), fuzzy logic utilization (4.3.3), and commanded output (4.3.4). These models are discussed in the following sections.

### 4.3.1 Reference Command Block

The reference command block shown in Figure 4.8 takes three inputs. The first is a "Start" command. When the "Start" constant block is "1", the fans are free to act. A "goto" tag block sends the command to each fan block to enable them.

The second input is the actual reference command and is used to calculate reference error to the controller and is output to the summing block.

The third input is the deceleration setpoint. From the model perspective, this is the point where the controller will become active. This setpoint is subtracted from the reference to provide proper scaling to the controller once it becomes active. This value is output to the scaling block.

### 4.3.2 Feedback

Feedback is provided by the position sensor. The position sensor sends back a millivolt reading to the Arduino. An analog block from within the MATLAB library is used to report the feedback of the position to the model and is shown in Figure 4.9. The signal is converted to a type double and sent through the linear conversion equation obtained from the calibration of the sensor. This reading is subtracted from the reference command to give a position error. The position error is then scaled and sent to the fuzzy controller.

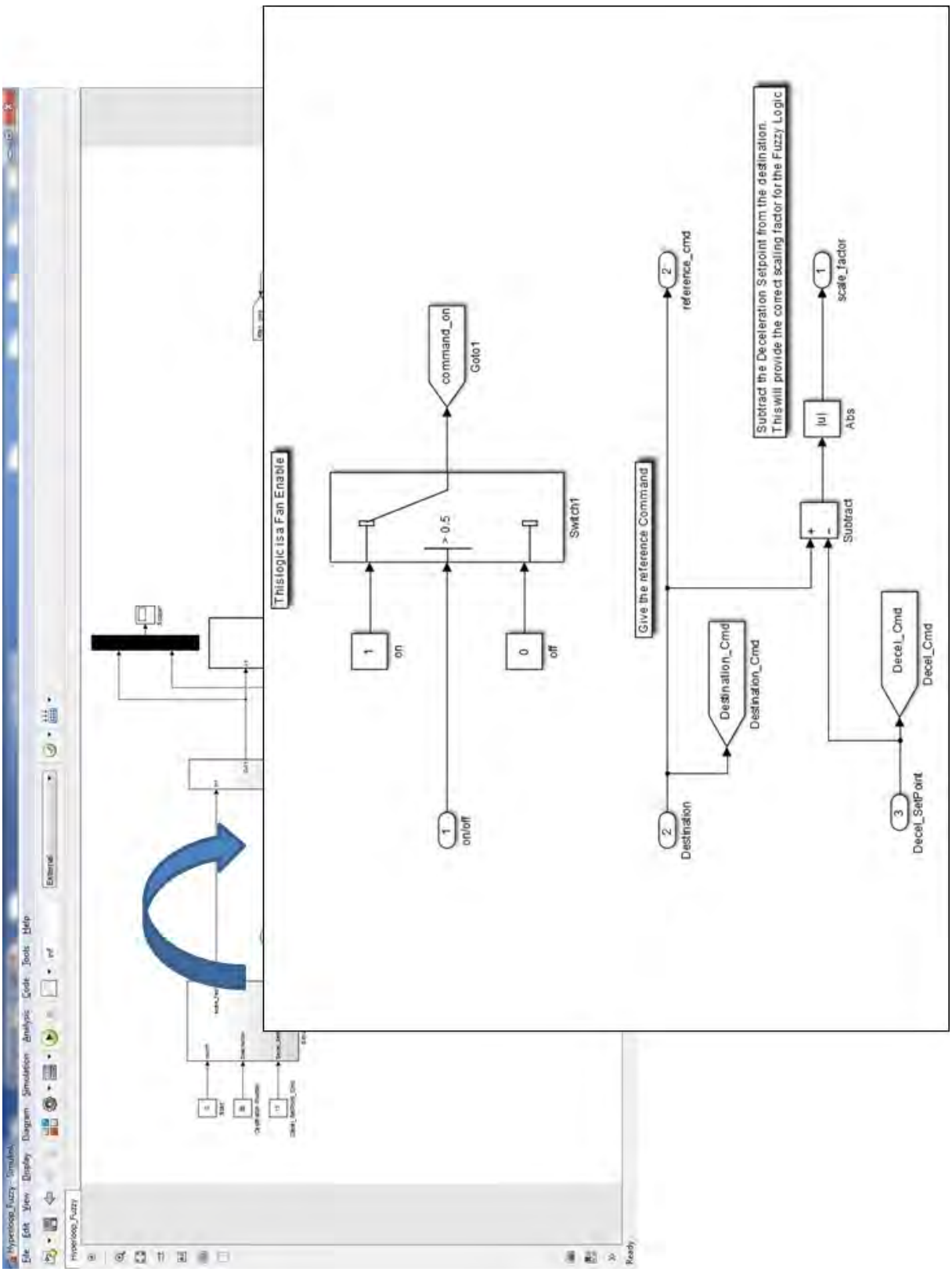



Figure 4.8: Inside the Reference Command Block.

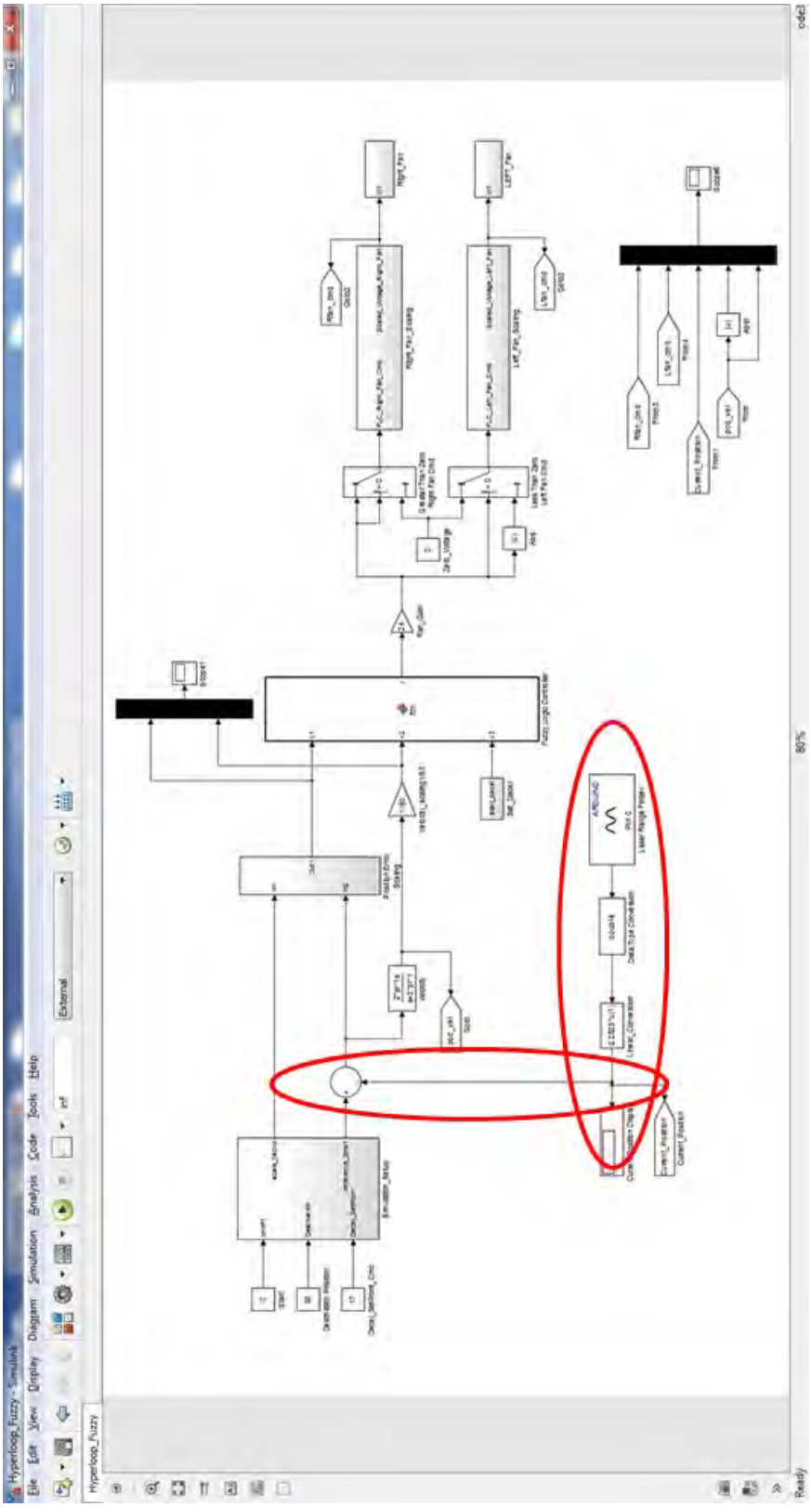


Figure 4.9: Position Sensor - Feedback.

### 4.3.3 Fuzzy Logic Controller

Inside the reference command block (section 4.3.1) the scaling factor is determined. In this section the scaling factor is simply utilized and is shown in Figure 4.10. Note that until the pod has reaches the deceleration set point, a feed forward command is given to accelerate the pod. Once the pod has reached the deceleration set point (the controller becomes active), the scaling section will provide a true scaled input into the controller. In Figure 4.11, the yellow line represents the output scaling for the position error. The blue line is the output of the fuzzy controller to the fan. When the fuzzy controller becomes active, a command of -1 is sent to the left fan by the controller. At this same time the position of the pod (scaled) is 1. Near the end of the stop, the position error has reached "0" as well as the command from the fuzzy controller. Thus, we have scaled the position error from 1 to 0 to the fuzzy controller.

The logic to determine when the controller becomes active is shown in Figure 4.12. The deceleration set point and the current position of the pod are subtracted. When the result is zero (0), a command of "1" is sent to the fuzzy control block. Within the fuzzy control block, logic is placed to activate the fuzzy controller. A switch box is used in conjunction with a memory block to hold the controller in the active state. Output tags are placed within this block to pass active values to the fan logic.

Input into the fuzzy controller is a scaled position error, a scaled change in position error and a decleration reference point (which is the logic for activation of the fuzzy controller). The Fuzzy Controller was placed inside a MATLAB function block (see Figure 4.13) and the code can be viewed in section G.0.1. The fuzzy controller passes a scaled output (-1 to 1) which is inturn is upscaled, mutliplying by 24 for a maximum of $\pm 24$VDC command to the fan. A negative voltage for the left fan and a positive voltage for the right. The "fuzzify" function called within the fuzzy controller can be viewed in section G.0.2.

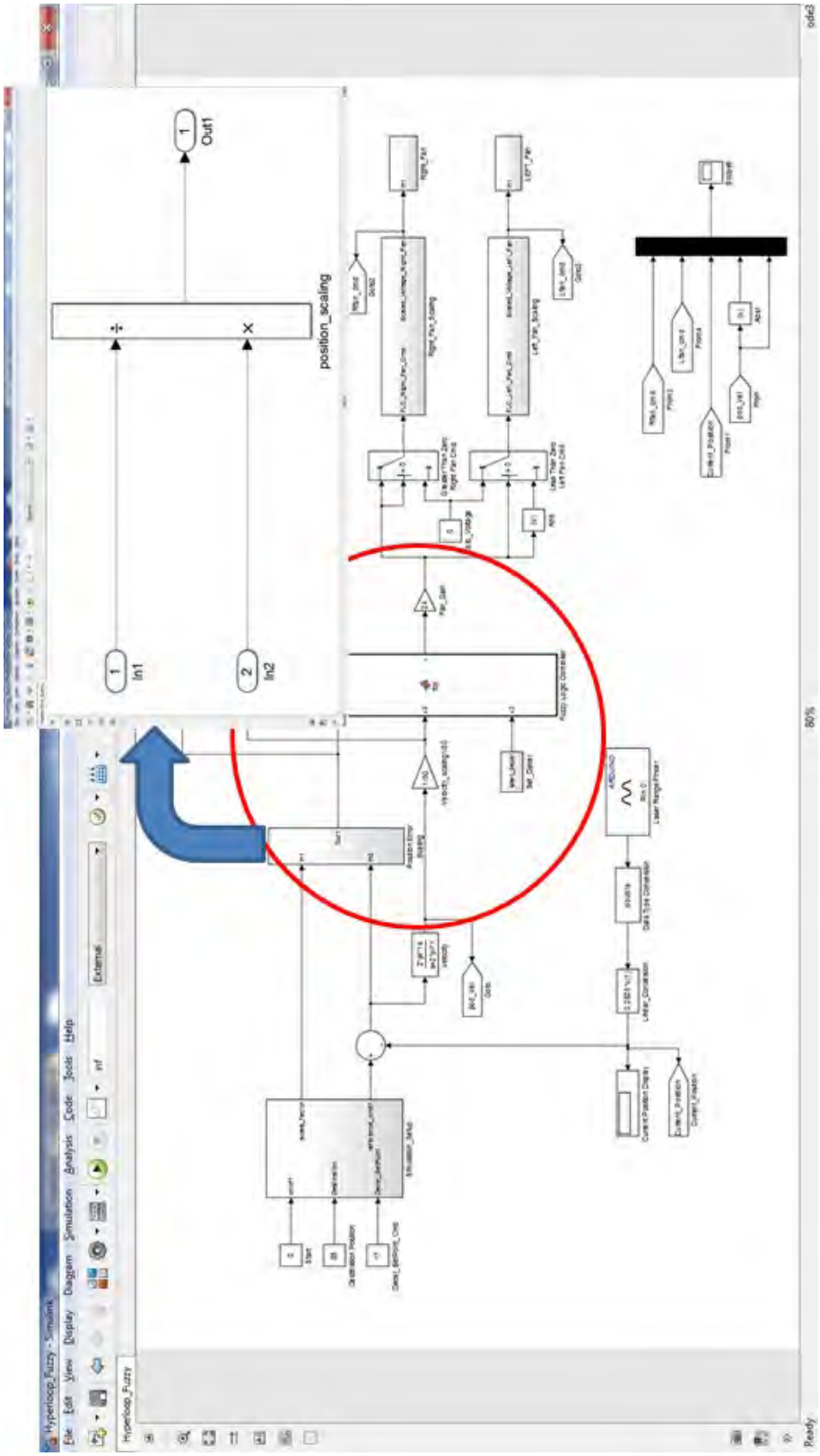



Figure 4.10: Fuzzy Logic Scaling.


Figure 4.11: Scaling Confirmation.

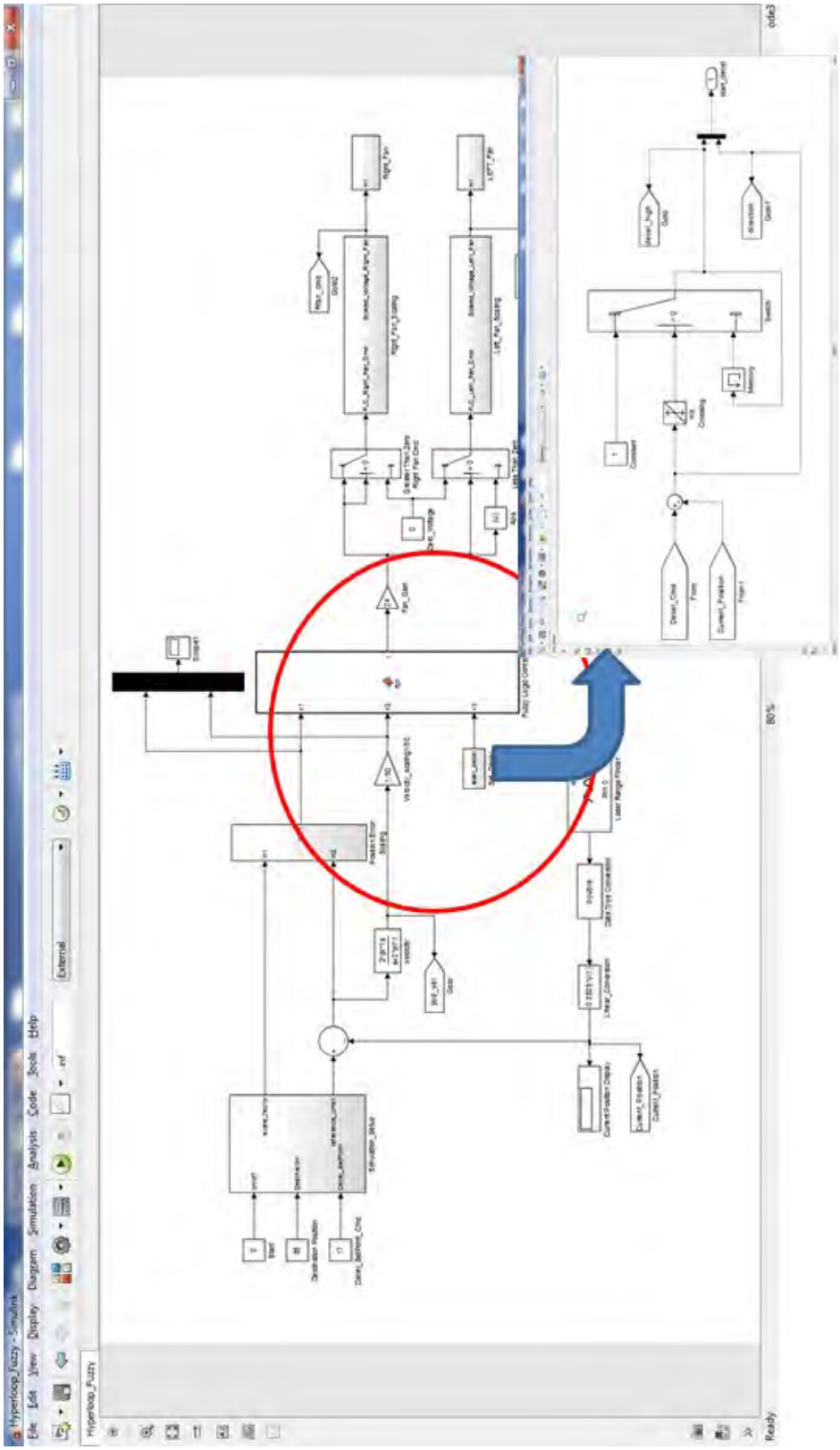


Figure 4.12: Deceleration Logic.

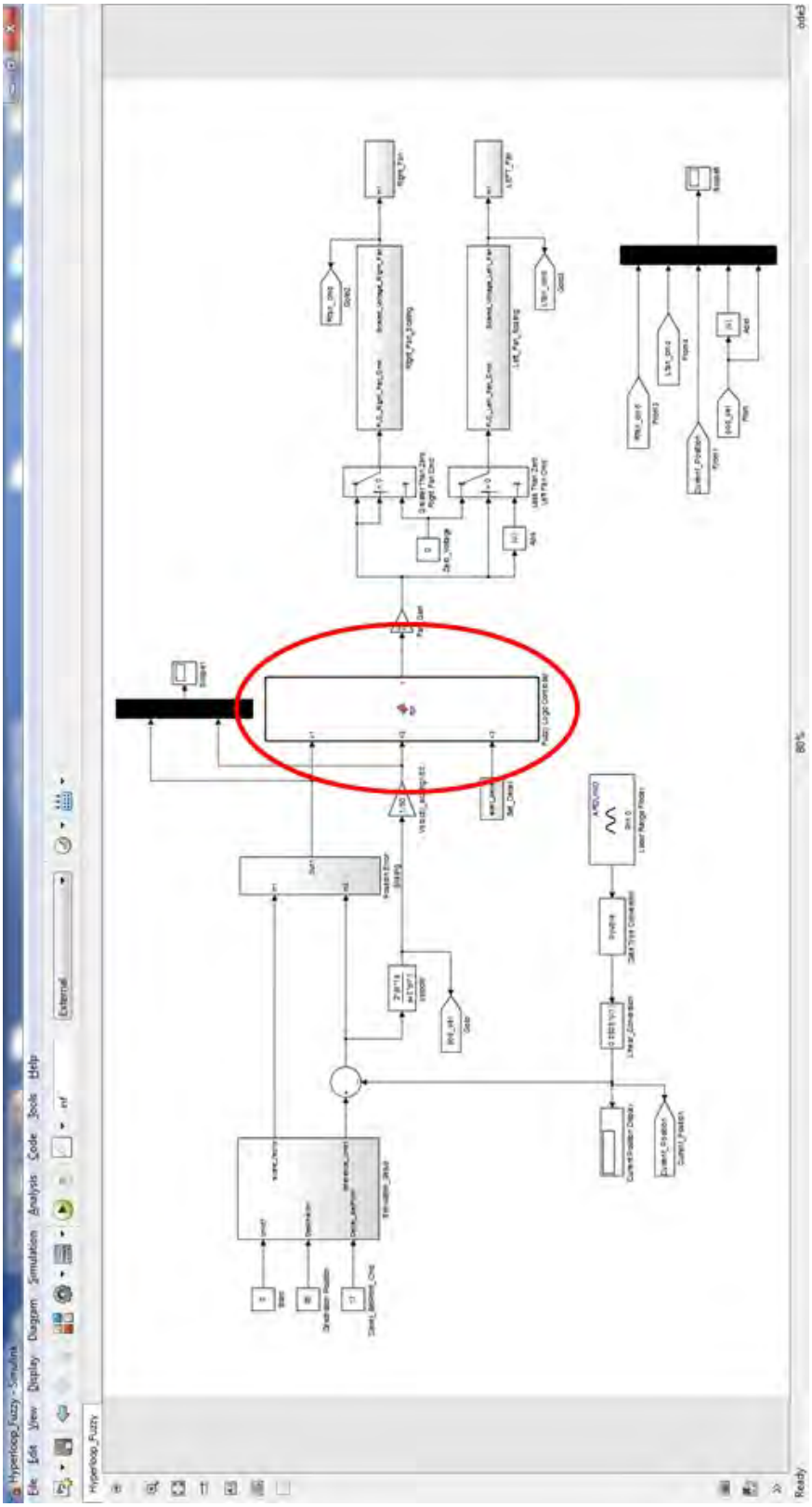


Figure 4.13: Fuzzy Control Block.

### 4.3.4 Commanded Output

The 0 - 24VDC commanded output from the fuzzy controller is passed through a swtich block. Logic from the deleration set point block directs the signal flow to the proper fan. Before this happens however, logic was placed which determines whether the pod has reached it destination and can be viewed in section 4.14.

Inside the fan logic block shown in Figure 4.15, A "From" tag passed from the reference block enables the fan. If the "command_on" is "1" the fan is enabled and the command from the fuzzy logic block is passed through. The logic from the bottom left waits for the "decel" command to go high. Once this happens the velocity of the POD is monitored for a zero crossing. If the velocity crosses zero, the simulation is over regardless of the position the POD is currently at. This logic was put into placed to prevent POD movement from triggering the controller. Due to lack of ventilation, when the POD reaches its destination there is air pressure built up near the stopping point. Because the POD is light weight and on a near froctionless track, movement of the POD from the built up pressure would trigger the fuzzy controller (and thus the fans) after the vehicle had reached its end point. With the fan enabled, the signal from the fuzzy controller passes through to the fan block.

Figure 4.16 represents the logic and command sent to the fan. As stated before, isolation between each fan was imperative due to the large current spikes witnessed with both fans simutaneously active. In this illustration we see two PWM blocks. The top block operates the solid state relay to enable the fan. A positive value greater than zero enables the fan. The second PWM block sends a scaled value to the amplifier which commands the fan to the proper DC voltage. The signal from the fuzzy controller passes a 0-24VDC command which runs through a lookup table and outputs a 0-255 PWM signal to the amplifier. Calibration of the fans, for the lookup table can be seen in Figure 4.17.

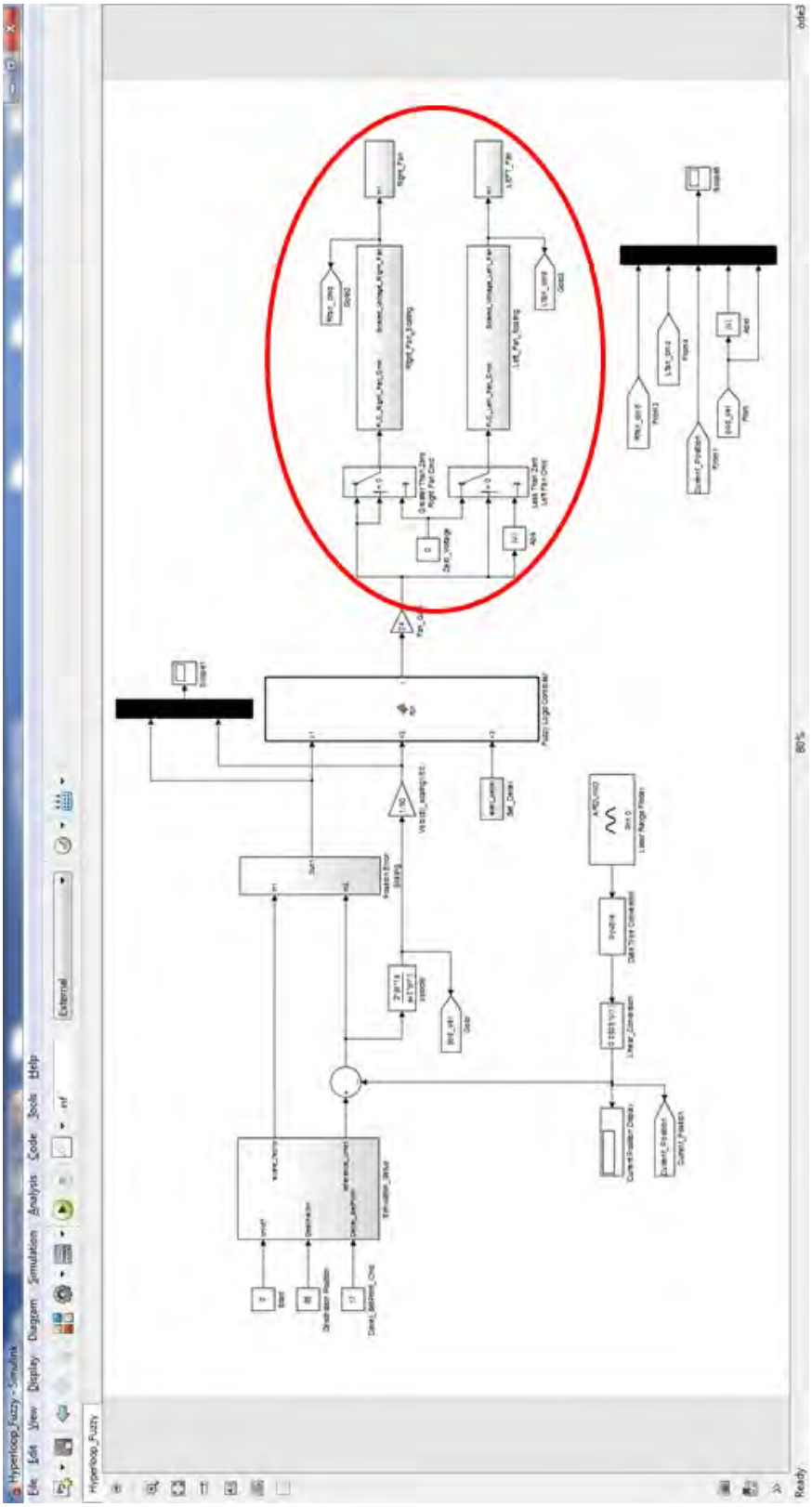


Figure 4.14: Commanded Output.

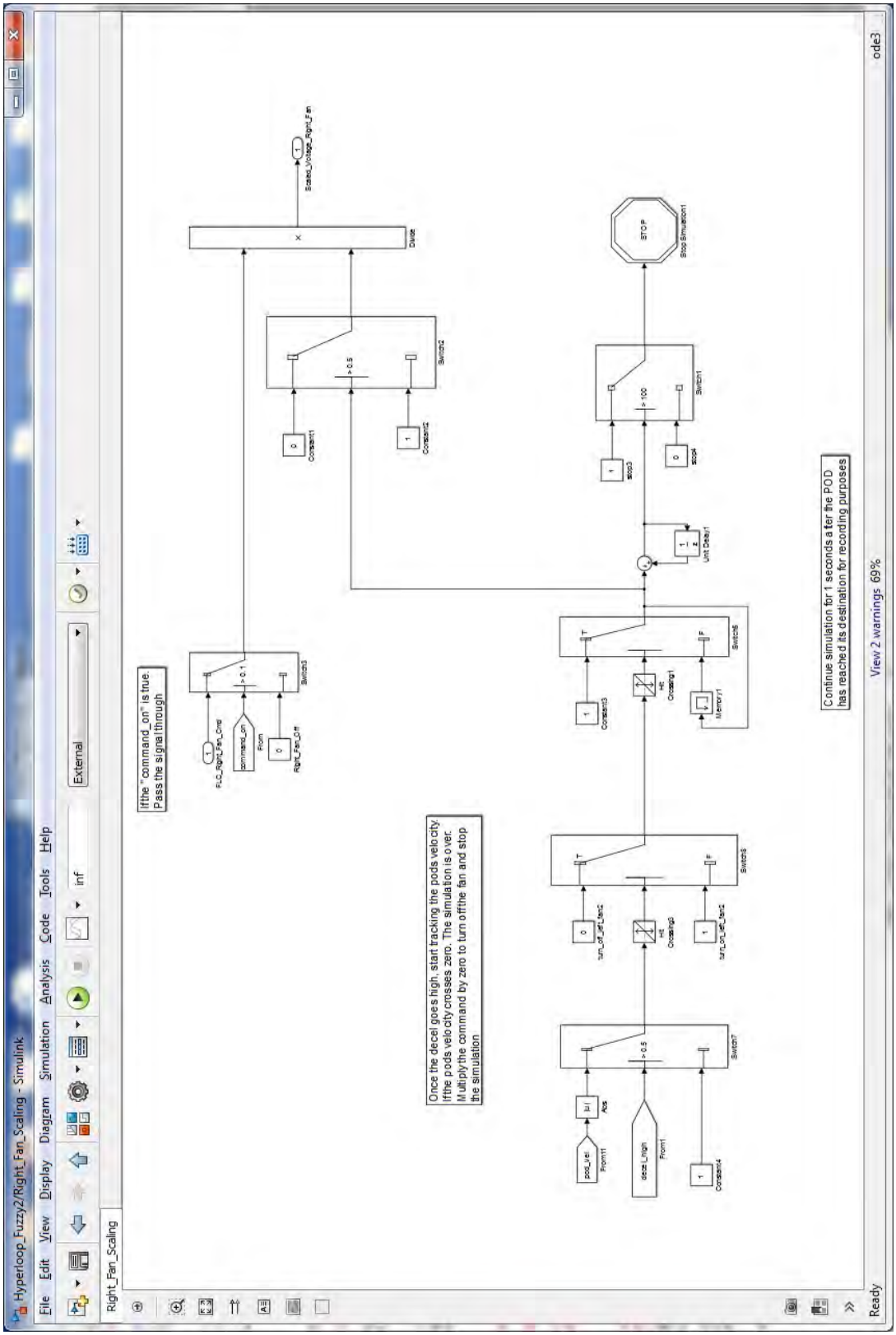



Figure 4.15: Fan Block Logic.

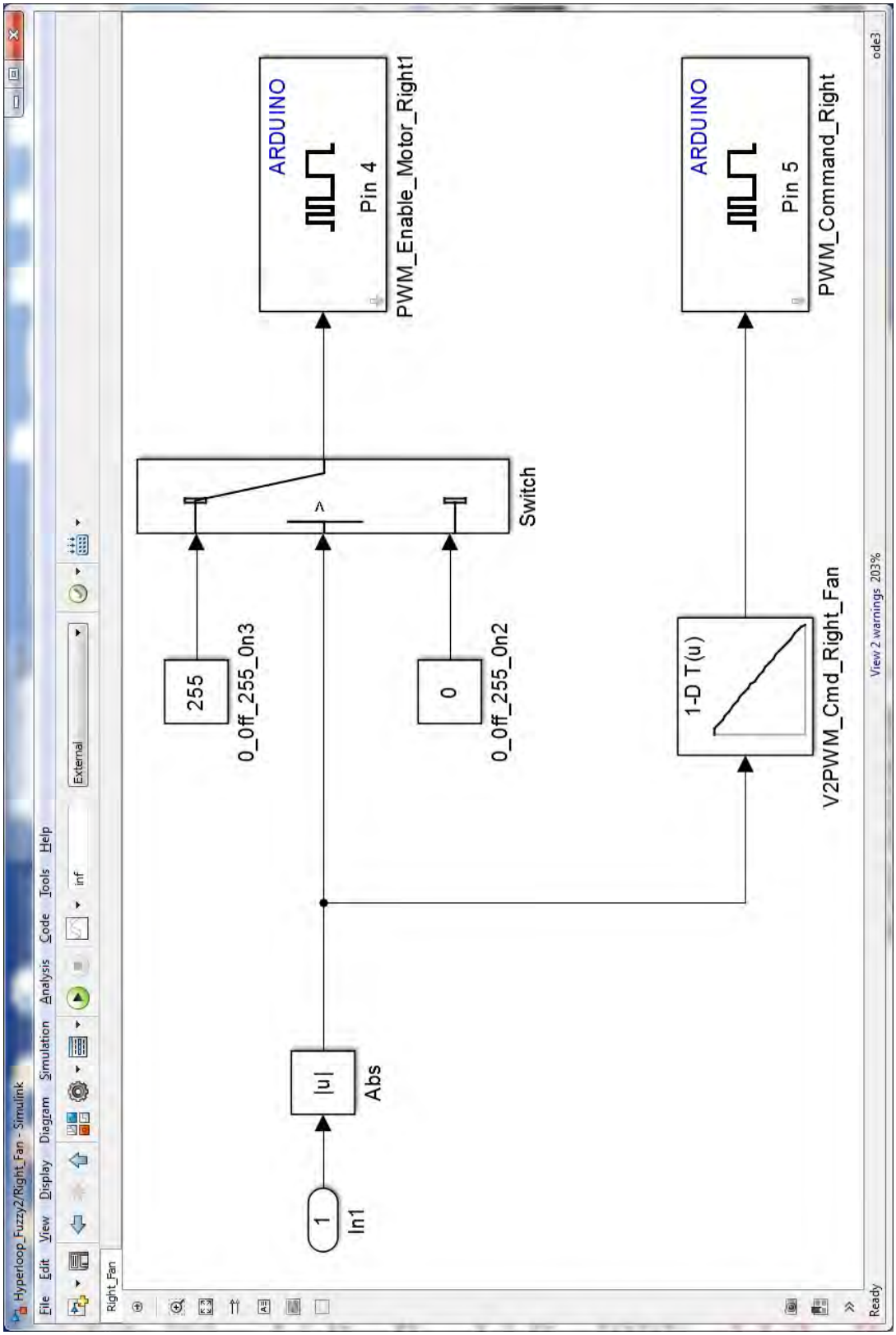



Figure 4.16: Fan Block.

| Right Fan | PWM | Applied Voltage | | Left Fan | PWM | Applied Voltage |
|---|---|---|---|---|---|---|
| | 255 | 1.1 | | | 255 | 1.2 |
| | 250 | 2.5 | | | 250 | 2 |
| | 240 | 3.5 | | | 240 | 30 |
| | 230 | 4.2 | | | 230 | 3.7 |
| | 220 | 5.2 | | | 220 | 4.7 |
| | 210 | 6.1 | | | 210 | 5.6 |
| | 200 | 7.1 | | | 200 | 6.5 |
| | 190 | 8 | | | 190 | 7.4 |
| | 180 | 8.9 | | | 180 | 8.3 |
| | 170 | 9.8 | | | 170 | 9.2 |
| | 160 | 10.6 | | | 160 | 10.1 |
| | 150 | 11.5 | | | 150 | 10.8 |
| | 140 | 12.3 | | | 140 | 11.9 |
| | 130 | 13.2 | | | 130 | 12.8 |
| | 120 | 14.1 | | | 120 | 13.5 |
| | 110 | 15 | | | 110 | 14.3 |
| | 100 | 15.9 | | | 100 | 15.4 |
| | 90 | 16.8 | | | 90 | 16.3 |
| | 80 | 17.7 | | | 80 | 17.2 |
| | 70 | 18.6 | | | 70 | 18.1 |
| | 60 | 19.4 | | | 60 | 19 |
| | 50 | 20.3 | | | 50 | 19.8 |
| | 40 | 21 | | | 40 | 20.6 |
| | 30 | 22 | | | 30 | 21.5 |
| | 20 | 22.9 | | | 20 | 22.4 |
| | 10 | 23.6 | | | 10 | 23.4 |
| | 0 | 23.8 | | | 0 | 23.8 |

Figure 4.17: Fan Calibration.

Fan calibration was carried out using the open loop model. The PWM signal for each fan was sequentially varied and the voltage output from the amplifiers was measured. The use of a MATLAB 1-D lookup table outputs the correct PWM (0-255) command for a given voltage command from the fuzzy controller.

## 4.4 Results

Two (2) simulations were performed sequentially for each test moving the POD in a right to left and left to right manner. For each stop, the POD was placed on the track farthest away from the destination to use the maximum available track. Each simulation was run using MATLAB script for repeatability and accuracy of testing. The scripts can be viewed in Appendix G.0.3 and G.0.4. Analysis scripts were also written to break down each simulated test run. The scripts can be viewed in Appendix G.0.5 and G.0.6.

For the first test, the pod was placed to the far right side of the track closest to the position sensor. A destination position of 85 was commanded with a deceleration set point at 16.5. The results are shown in Figures 4.18 and 4.19

For the second test, the pod was placed to the far left side of the track furthest away from the position sensor. A destination position of 10 was commanded with a deceleration set point at 78. The results are shown in Figures 4.20 and 4.21.

For the test results figures that follow, the legend on the figures are defined as follows:

RFv - Right Fan Voltage Command

LFv - Left Fan Voltage Command

Ppos - Pod position

Pvel - Pod velocity

PPvel - absolute velocity

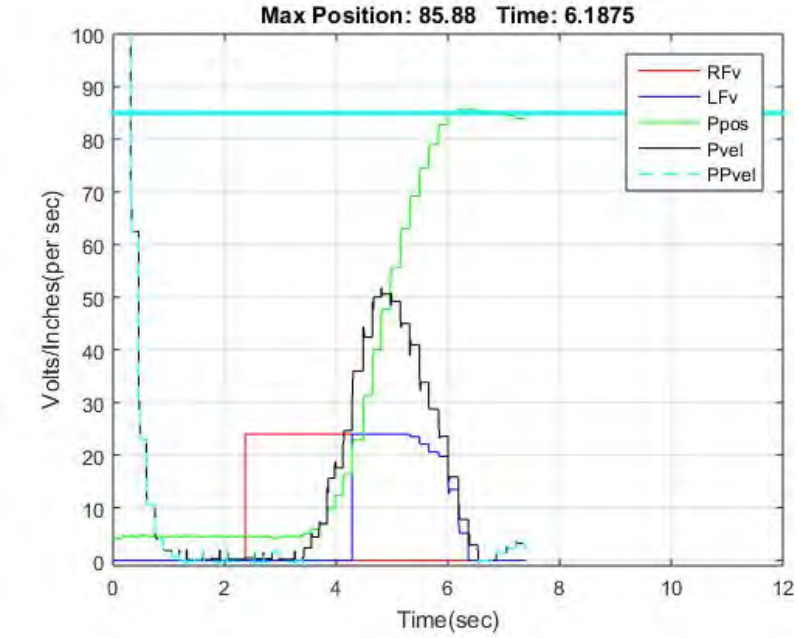



Figure 4.18: Closed Loop - Right to Left - 1.

For this simulation, 24VDC was commanded to the right fan at 2.364 seconds into the test. A notable velocity change was seen at 3.400 seconds. The right fan continued to run at the 24VDC command until 4.281 seconds into the test, at which time the POD reached the deceleration set point of 16.5" and the fuzzy controller became active commanding the left fan on full at 24VDC. The velocity of the POD continued to gain in speed until its maximum velocity at 4.805 seconds. The left fan stayed on at 24VDC until the controller began scaling back the voltage at 5.312 seconds. The controller continued dropping the voltage on the left fan, as it tracked the POD velocity, until 0VDC was commanded at 6.365 seconds. The POD reached the destination position with a .88" overshoot at 6.187 seconds and the POD came to a complete resting point with zero velocity a 6.541 seconds at which point the simulation was over.

The right fan was on for a total of 1.916 seconds, the left fan was on a total of 2.083 seconds and the POD reached a maximum velocity of 51.739 inches per second.

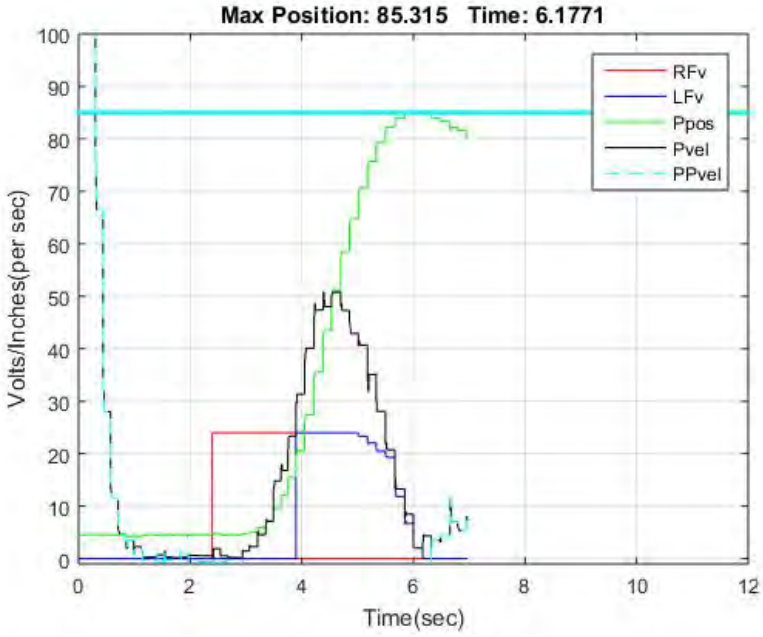



Figure 4.19: Closed Loop - Right to Left - 2.

For this simulation, 24VDC was commanded to the right fan at 2.396 seconds into the test. A notable velocity change was seen at 3.208 seconds. The right fan continued to run at the 24VDC command until 3.895 seconds into the test, at which time the POD reached the deceleration set point of 16.5" and the fuzzy controller became active commanding the left fan on full at 24VDC. The velocity of the POD continued to gain in speed until its maximum velocity at 4.385 seconds. The left fan stayed on at 24VDC until the controller began scaling back the voltage at 4.875 seconds. The controller continued dropping the voltage on the left fan, as it tracked the POD velocity, until 0VDC was commanded at 6.166 seconds. The POD reached the destination position with a .315" overshoot at 6.177 seconds and the POD came to a complete resting point with zero velocity a 6.188 seconds at which point the simulation was over.

The right fan was on for a total of 1.496 seconds, the left fan was on a total of 2.280 seconds and the POD reached a maximum velocity of 50.810 inches per second.

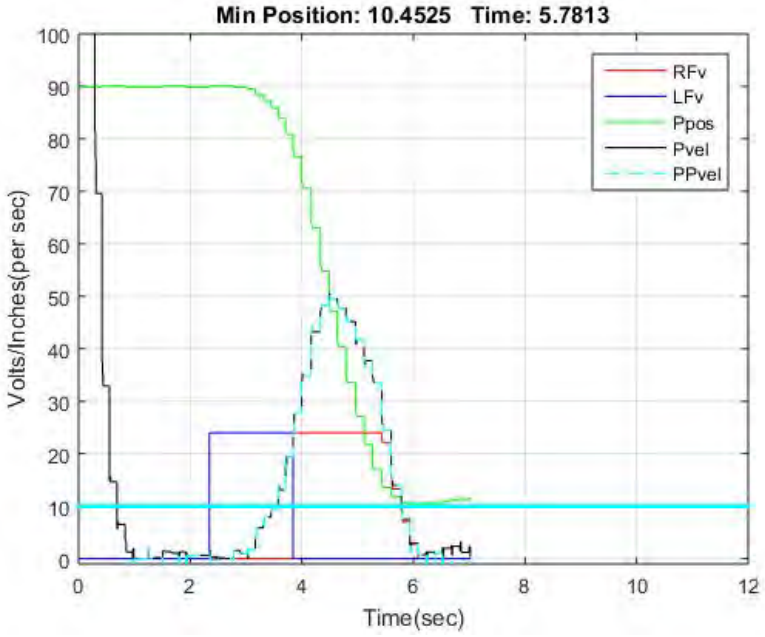



Figure 4.20: Closed Loop - Left to Right - 1.

For this simulation, 24VDC was commanded to the left fan at 2.343 seconds into the test. A notable velocity change was seen at 3.166 seconds. The left fan continued to run at the 24VDC command until 3.729 seconds into the test, at which time the POD reached the deceleration set point of 78" and the fuzzy controller became active commanding the left fan on full at 24VDC. The velocity of the POD continued to gain in speed until its maximum velocity at 4.489 seconds. The right fan stayed on at 24VDC until the controller began scaling back the voltage at 5.437 seconds. The controller continued dropping the voltage on the right fan, as it tracked the POD velocity, until 0VDC was commanded at 5.958 seconds. The POD reached the destination position with a .453" overshoot at 5.781 seconds and the POD came to a complete resting point with zero velocity a 6.105 seconds at which point the simulation was over.

The left fan was on for a total of 1.385 seconds, the right fan was on a total of 2.115 seconds and the POD reached a maximum velocity of 50.516 inches per second.

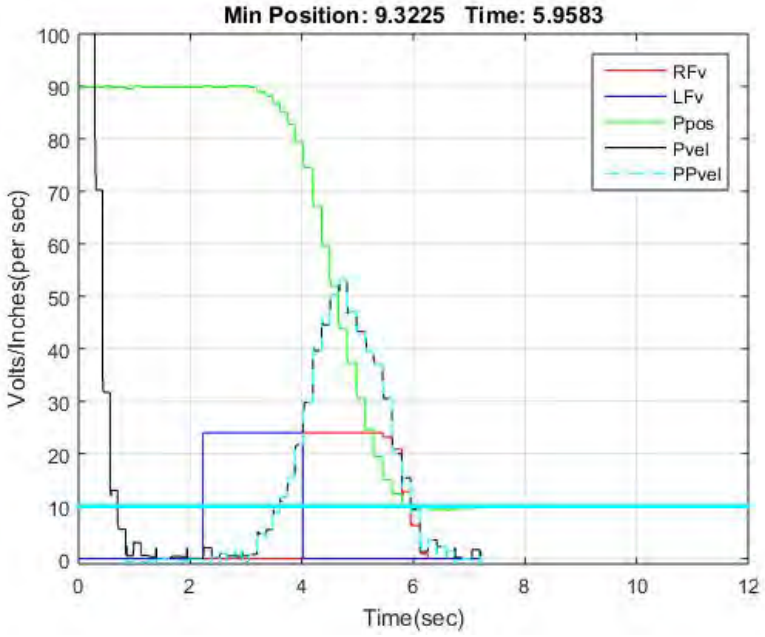



Figure 4.21: Closed Loop - Left to Right - 2.

For this simulation, 24VDC was commanded to the left fan at 2.229 seconds into the test. A notable velocity change was seen at 3.198 seconds. The left fan continued to run at the 24VDC command until 3.896 seconds into the test, at which time the POD reached the deceleration set point of 78" and the fuzzy controller became active commanding the left fan on full at 24VDC. The velocity of the POD continued to gain in speed until its maximum velocity at 4.677 seconds. The right fan stayed on at 24VDC until the controller began scaling back the voltage at 5.447 seconds. The controller continued dropping the voltage on the right fan, as it tracked the POD velocity, until 0VDC was commanded at 6.135 seconds. The POD reached the destination position with a .677" undershoot at 5.958 seconds and the POD came to a complete resting point with zero velocity a 6.135 seconds at which point the simulation was over.

The left fan was on for a total of 1.667 seconds, the right fan was on a total of 2.114 seconds and the POD reached a maximum velocity of 53.263 inches per second.

With the fuzzy controller tuned to an acceptable setting, other simulated test were run and the analysis scripts captured the data. For this "Right to Left" simulation the destination position was changed to 88 and the deceleration set point was set to 17.5. The simulation was ran and the results can be seen in Figure 4.22.

| Right Fan On | Right Fan Off | Right Fan Total Time |
|---|---|---|
| 2.34375 | 3.71875 | 1.375 |
| | | |
| Left Fan On | Left Fan Off | Left Fan Total Time |
| 3.833333492 | 6.239583492 | 2.40625 |
| | | |
| Max. Velocity | Max. Velocity Time | |
| 53.37060547 | 4.5 | |
| | | |
| Max. Position | Max. Position Time | Beyond Target Position |
| 88.42250061 | 5.916666985 | 0.42250061 |

Figure 4.22: Closed Loop - Right To Left - Dest. 88 Decel. 17.5.

From the figure above, there is a slight increase from the previous test in the maximum velocity. The right fan (which was used to accelerate the POD) was on a total of 1.375 seconds and the left fan (deceleration) was on longer at 2.406 seconds. The overshoot in this case was still under half an inche at .4225 inches.

Next, adjustments were made to the script which sends the POD in a "Left to Right" direction. The destination position was changed to 7 and the deceleration set point was adjusted to 77. The simulation was ran and the results can be seen in Figure 4.23.

| Right Fan On | Right Fan Off | Right Fan Total Time |
|---|---|---|
| 4 | 6.145833492 | 2.145833492 |
| | | |
| Left Fan On | Left Fan Off | Left Fan Total Time |
| 2.333333492 | 3.989583492 | 1.65625 |
| | | |
| Max. Velocity | Max. Velocity Time | |
| 52.79069519 | 4.604166985 | |
| | | |
| Min. Position | Min. Position Time | Beyond Target Position |
| 5.932499886 | 6.135416985 | 1.067500114 |

Figure 4.23: Closed Loop - Left To Right - Dest. 7 Decel. 77.

From the figure above, there is a slight increase from the previous test in the maximum velocity. The left fan (which was used to accelerate the POD) was on a total of 1.656 seconds and the right fan (deceleration) was on longer at 2.145 seconds. The overshoot in this case was longer at 1.065 inches.

# Conclusion

## 5.1 Conclusion

This thesis has shown that Fuzzy Logic position control could be used as a possible means to bring Hyperloop pods to their destination points. The Fuzzy logic controller for this thesis proved to show acceptable stopping point positions using traceable control methodologies.

It is important to keep in mind the nonlinearities of the system that the controller was operating on. Air is compressible, and it was not possible to measure the air leakage between the pod and the tunnel as it moved down the track. Moreover, the periodic friction on the keel of the pod against the track changed for each simulated stop. Venting the compressed air near stopping positions was not incorporated and therefore the system built was based on an over damped system.

Given these conditions, the controller was able to produce acceptable results and react to a system with slow response. Final stopping distance variances across each simulated test run were viewed as reasonable and acceptable for this thesis given the nonlinearities of the system. Once the fuzzy controller was implemented and tuned, adjustments of the stopping positions were simply applied to the parameters of the model (i.e. destination and deceleration set point). In short, the fuzzy controller did the rest. Each simulated test run came within an acceptable range of the desired postion.

## 5.2 Future Work

Possible improvements to the system presented in this thesis could be improved on in the following areas. Incorporating pressure sensors for feedback would allow tracking of the applied force acting on the pod. Venting of air pressure at each end of the track or perhaps in strategic positions along the length of the track would allow pressure in front of the pod to be reduced, creating a greater pressure differential to the front and back surface areas of the pod. Perhaps the most significant upgrade to the system would be to use fans that allow polarity changes.

Fans that could push or draw air as commanded would allow the incorporation of an integrator into the system controls. Using fans with incorporated resolvers would allow fan feedback, which would provide information for an inner closed-loop around the fans. The system presented in this thesis relied on a light-weight pod and an oversized fan. Future work may include a heavier pod with an undersized fan which would allow the linear momentum (mass * velocity) of the pod to compress the air even further at the destination points rather than rely totally on fan velocity to stop the pod. However, this would require longer track length for testing. Other avenues to consider would be on the use of air pumps to move outside air into the tunnel and produce force which would move the pod. Upgrading the position sensor and processing board could possibly allow for cleaner signals, thus opening the door for tracking velocity and accelerations by the controller.

# Bibliography


[1] Elon Musk, CEO SpaceX. *Hyperloop Alpha*.
http://www.spacex.com/sites/spacex/files/hyperloop_alpha-20130812.pdf

[2] Arduino. *Arduino Mega Overview*.
https://www.arduino.cc/en/Main/ArduinoBoardMega2560

[3] Mathworks Inc., *What is MATLAB*.
http://www.mathworks.com/

[4] Mathworks Inc., *Mathworks Products and Services*.
http://www.mathworks.com/

[5] Mathworks Inc., *Simulation and Model-Based Design*
http://www.mathworks.com/products/simulink/

[6] Joseph Brennan, *Beach Pneumatic*
*Alfred Beach's Pneumatic Subway and the beginnings of rapid transit in New York*
http://www.columbia.edu/~brennan/beach/

[7] Joseph Brennan, *Atmospheric Railways in England*
*Alfred Beach's Pneumatic Subway and the beginnings of rapid transit in New York*
http://www.columbia.edu/~brennan/beach/chapter2.html

[8] Joseph Brennan, *Atmospheric Railways in England*

*Alfred Beach's Pneumatic Subway and the beginnings of rapid transit in New York*

http://www.columbia.edu/~brennan/beach/chapter6.html

[9] Paul Hernandez, Rita Evans, Rahul Kamath, Sean Mooney,

*Transportation Futuristics*

http://www.lib.berkeley.edu/news_events/futuristics/pt/

[10] Lofti A. Zadeh, *Outline of a New Approach to the Analysis of Complex Systems and Decision Processes*

*IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, VOL. SMC-3, NO.1, JANUARY 1973* https://people.eecs.berkeley.edu/~zadeh/papers/1973-Outline%20of%20a%20New%20Approach%20to%20the%20Analysis%20of%20Complex%20Systems%20and%20Decision%20Processes.pdf

[11] Smithsonian National Postal Museum, *Pneumatic Tube Mail*

http://postalmuseum.si.edu/exhibits/current/customers-and-communities/serving-the-cities/city-free-delivery/pneumatic-tube-mail.html

[12] Smithsonian National Postal Museum, *Pneumatic Tube Mail*

http://postalmuseum.si.edu/exhibits/current/customers-and-communities/serving-the-cities/city-free-delivery/pneumatic-tube-mail.html

[13] Nancy A. Pope, Smithsonian National Postal Museum, *Pneumatic Tube Mail*, 2005

http://postalmuseum.si.edu/collections/object-spotlight/pneumatic-mail.html

[14] Michael del Castillo, Upstart Business Journal Technology & Innovation Editor, *Hyperloop is open source: 5 ways to get involved*, August 13, 2013, 12:49pm EDT
http://upstart.bizjournals.com/news/technology/2013/08/13/elon-musk-wants-help-with-hyperloop.html?page=all

[15] SpaceX. *The Official SpaceX Hyperloop POD Competition*.
http://www.spacex.com/hyperloop

[16] Andrew J . Hawkins. *The Verge*. June 18, 2016 03:40 pm
http://www.theverge.com/2016/6/18/11965354/hyperloop-pod-competition-elon-musk-spacex-team-design

[17] SpaceX. *SpaceX Hyperloop Test-Track Specifications*. June 18, 2016 03:40 pm
https://badgerloop.com/documents/TubeSpecs.pdf

[18] Hyperloop One. *Be anywhere. Move anything.Connect everyone.*.
https://hyperloop-one.com/

[19] Rick Stella. *Digital Trends*, May 14, 2016 3:00 AM
*The Hyperloop is real, and we watched the first test in the Nevada desert*.
http://www.digitaltrends.com/cool-tech/hyperloop-one-debuts-its-hyperloop-system/

[20] Andrew J . Hawkins. *The Verge*. May 9, 2016, 9:00am
*Hyperloop Transportation says it will use a âĂŸcheaper, saferâĂŹ form of magnetic levitation*.
http://www.theverge.com/2016/5/9/11636460/hyperloop-transportation-passive-magnetic-levitation-inductrack-ric

[21] Online Vendor *Digi-Key Electronics*. Friday, September 16, 2016 8:25:49 AM
http://www.digikey.com/?&WT.srch=1&gclid=CLO59qbwk88CFZElgQod7ysDwg

[22] Mathworks Inc., *Tune and Monitor Model Running on Arduino Mega 2560 Hardware*.
http://www.mathworks.com/help/supportpkg/arduino/ug/tune-and-monitor-model-running-on-arduino-mega-2560-hardware.html

[23] Knuth: Computers and Typesetting,
http://www-cs-faculty.stanford.edu/ uno/abcde.html

[24] Sameep Singh, *Implementation of a Fuzzy Logic Controller on an FPGA using VHDL* MSEE, Wright State University, 2002, Page 26 - Figure 2.6.

[25] Sameep Singh, *Implementation of a Fuzzy Logic Controller on an FPGA using VHDL* MSEE, Wright State University, 2002, Page 27 - Section 2.6.3.

# Appendix A

# Preliminary Track Construction

### A.0.1 First Preliminary Track

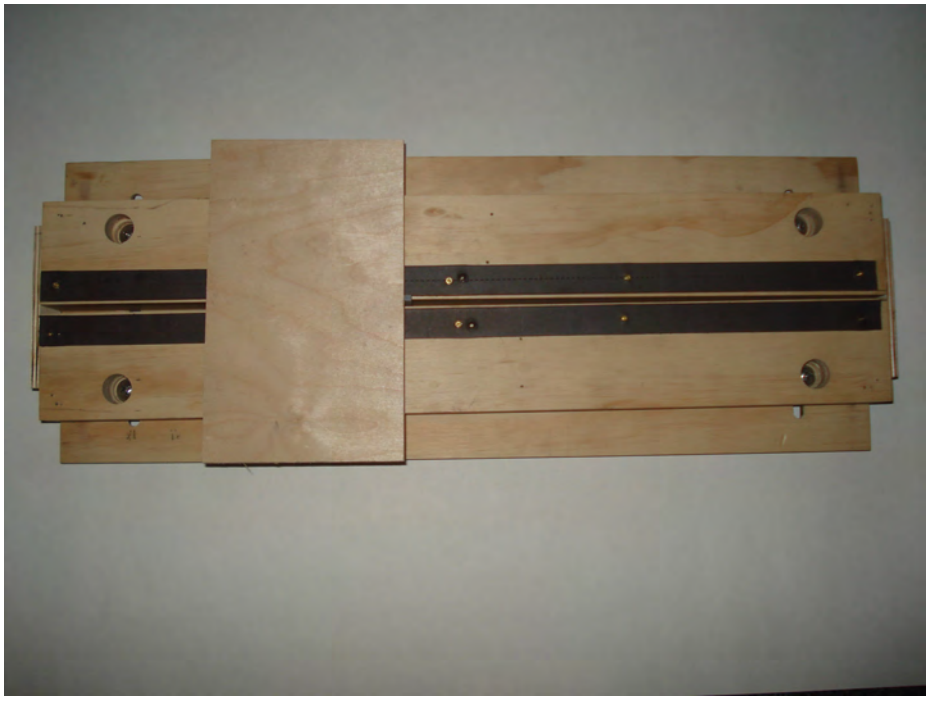


Figure A.1: First Preliminary Track.

### A.0.2 Second Preliminary Track

Figure A.2: Second Preliminary Track.

This is the second preliminary track cut from 2"x4" lumber.

### A.0.3 Second Preliminary Track

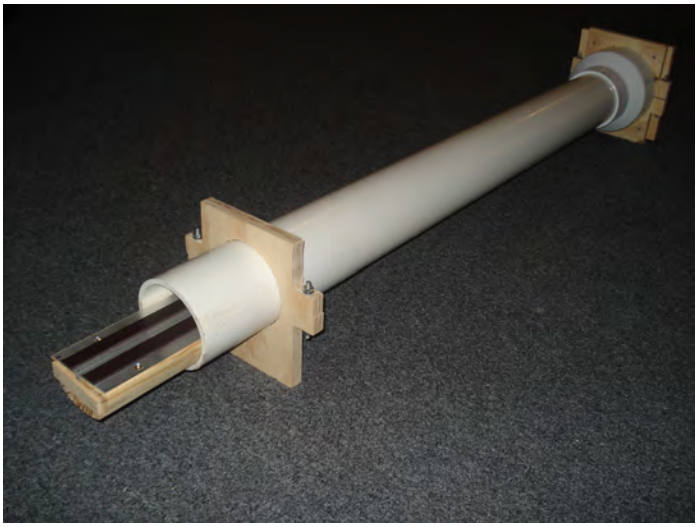


Figure A.3: Second Preliminary Track and tube.

### A.0.4 Second Preliminary Track

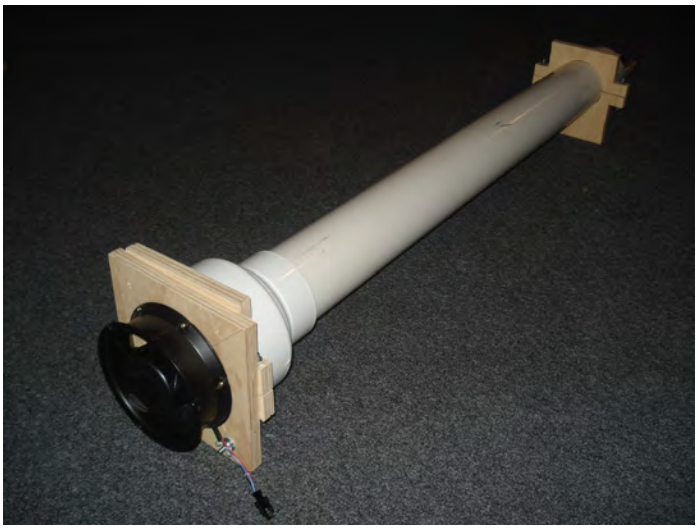


Figure A.4: Second Preliminary Track and fan.

# Appendix B

# DC Fan

### B.0.1 DC Fan Datasheet

**Motor:**

| | |
|---|---|
| Nominal Voltage | = 24 vdc |
| Operating Voltage Range | = 12 - 28 vdc |
| Running Current | = 1.00 amps |
| Locked Rotor Current | = 2.00 amps |
| Running Power | = 24 watts |
| Average Speed | = 3500 RPM |
| Air Flow | = 235 CFM |
| Bearings | = Ball |
| Acoustic | = 51.2 dBA |

**Construction:**

Venturi: Single Piece, Die-Cast Aluminum, Black
Propeller: Plastic, Black, UL 94V-0

**Agency Approvals:**

UL,CE

**Life Expectancy:**

This fan is designed for continuous duty life of 82,500 hours at 50°C.

Figure B.1: DC Fan Datasheet.

# Appendix C

# Handmade Power Supply

### C.0.1 Handmade Power Supply Front Panel

Figure C.1: Homemade Power Supply Front Panel.

The front panel interface allows (3) seperate supplies from (3) seperate linear regulating boards inside the box. Each supply can be hand adjusted from 0-30 VDC and can safely handle 1.5 amps. On/off switches allow the user to adjust voltage on the visual display before sending it to the front panel. The visual displays show both DC voltage supplied and current loads.

### C.0.2 Handmade Power Supply Inside - Cover Off

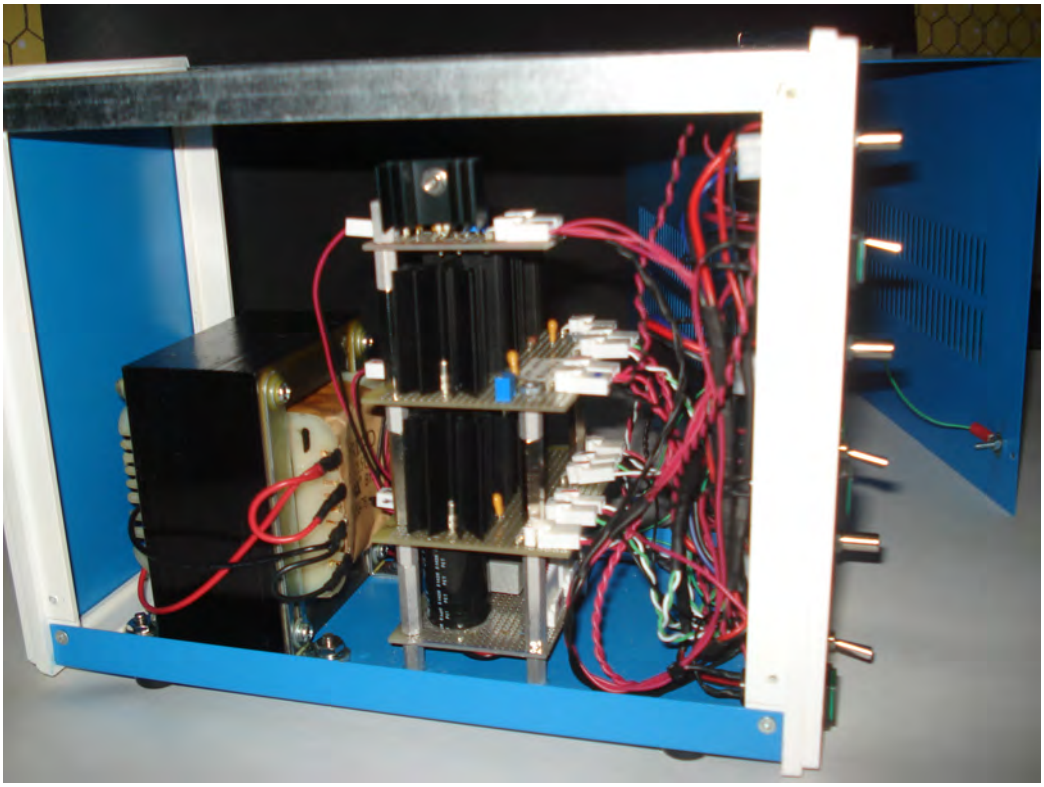
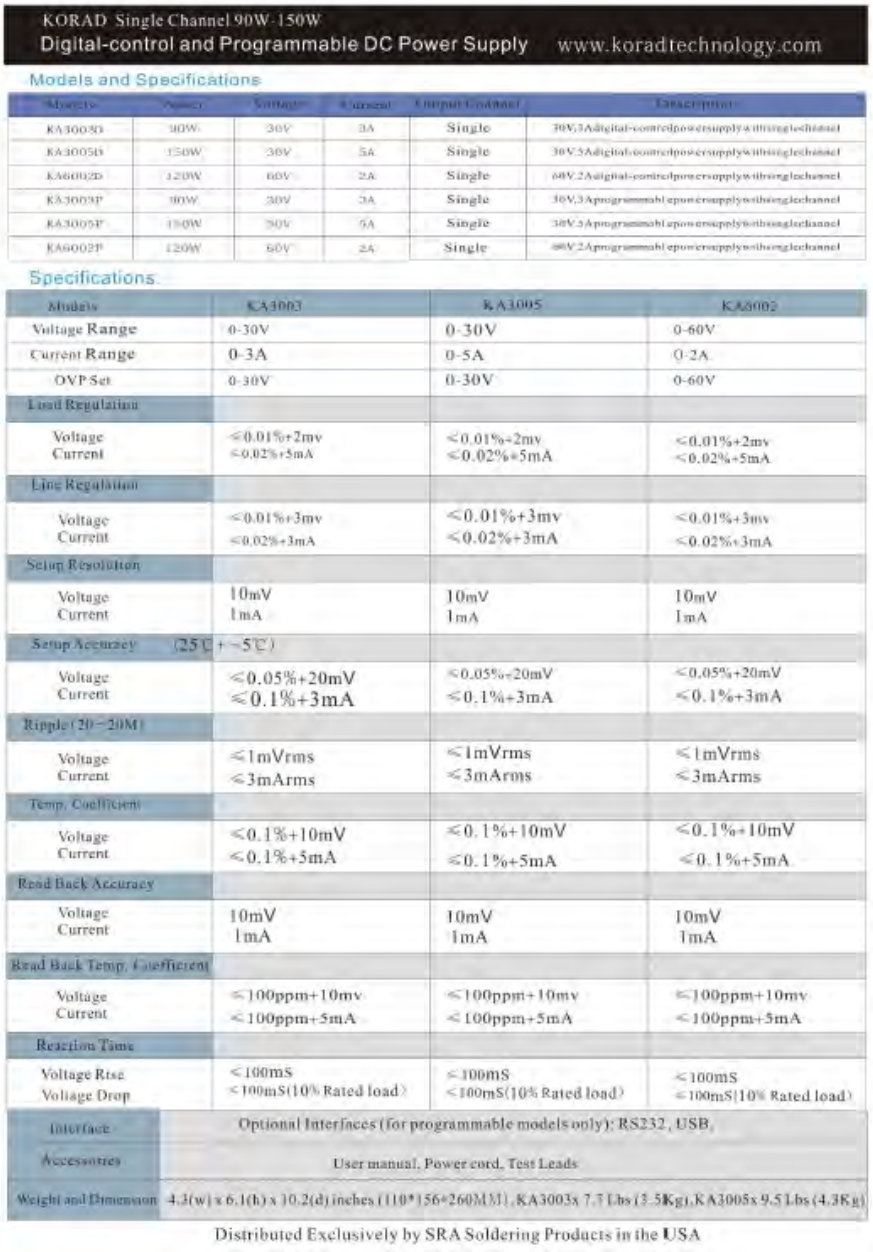


Figure C.2: Homemade Power Supply Inside - Cover off.

The transformer shown on the far left is used to step down the 110AC voltage coming into the box. The bottom layer of stacked boards contains a bridge rectifier and a large capacitor that changes the unregulated AC votage to DC. This DC voltage is then wired through Molex connectors to (3) linear voltage regulating boards. The regulating boards are capable of suppling 1.25 VDC - 37 VDC at 1.5Amps. Linear rotary potentiometers are used with the regulating boards to allow user adjustments for a desired output. Inline switches were incorporated to restrict power from reaching the front panel until selected.

## C.0.3 Power Supplies Korad KA3005D



KORAD Single Channel 90W-150W
Digital-control and Programmable DC Power Supply www.koradtechnology.com

Models and Specifications

| Models | Power | Voltage | Current | Output Channel | Description |
|---|---|---|---|---|---|
| KA3003D | 90W | 30V | 3A | Single | 30V,3A digital-control power supply with single channel |
| KA3005D | 150W | 30V | 5A | Single | 30V,5A digital-control power supply with single channel |
| KA6002D | 120W | 60V | 2A | Single | 60V,2A digital-control power supply with single channel |
| KA3003P | 90W | 30V | 3A | Single | 30V,3A programmable power supply with single channel |
| KA3005P | 150W | 30V | 5A | Single | 30V,5A programmable power supply with single channel |
| KA6002P | 120W | 60V | 2A | Single | 60V,2A programmable power supply with single channel |

Specifications

| Models | KA3003 | KA3005 | KA6002 |
|---|---|---|---|
| Voltage Range | 0-30V | 0-30V | 0-60V |
| Current Range | 0-3A | 0-5A | 0-2A |
| OVP Set | 0-30V | 0-30V | 0-60V |
| Load Regulation | | | |
| Voltage<br>Current | ≤0.01%+2mv<br>≤0.02%+5mA | ≤0.01%+2mv<br>≤0.02%+5mA | ≤0.01%+2mv<br>≤0.02%+5mA |
| Line Regulation | | | |
| Voltage<br>Current | ≤0.01%+3mv<br>≤0.02%+3mA | ≤0.01%+3mv<br>≤0.02%+3mA | ≤0.01%+3mv<br>≤0.02%+3mA |
| Setup Resolution | | | |
| Voltage<br>Current | 10mV<br>1mA | 10mV<br>1mA | 10mV<br>1mA |
| Setup Accuracy (25℃+-5℃) | | | |
| Voltage<br>Current | ≤0.05%+20mV<br>≤0.1%+3mA | ≤0.05%+20mV<br>≤0.1%+3mA | ≤0.05%+20mV<br>≤0.1%+3mA |
| Ripple(20~20M) | | | |
| Voltage<br>Current | ≤1mVrms<br>≤3mArms | ≤1mVrms<br>≤3mArms | ≤1mVrms<br>≤3mArms |
| Temp. Coefficient | | | |
| Voltage<br>Current | ≤0.1%+10mV<br>≤0.1%+5mA | ≤0.1%+10mV<br>≤0.1%+5mA | ≤0.1%+10mV<br>≤0.1%+5mA |
| Read Back Accuracy | | | |
| Voltage<br>Current | 10mV<br>1mA | 10mV<br>1mA | 10mV<br>1mA |
| Read Back Temp. Coefficient | | | |
| Voltage<br>Current | ≤100ppm+10mv<br>≤100ppm+5mA | ≤100ppm+10mv<br>≤100ppm+5mA | ≤100ppm+10mv<br>≤100ppm+5mA |
| Reaction Time | | | |
| Voltage Rise<br>Voltage Drop | ≤100mS<br>≤100mS(10% Rated load) | ≤100mS<br>≤100mS(10% Rated load) | ≤100mS<br>≤100mS(10% Rated load) |
| Interface | Optional Interfaces (for programmable models only): RS232, USB, | | |
| Accessories | User manual, Power cord, Test Leads | | |
| Weight and Dimension | 4.3(w) x 6.1(h) x 10.2(d) inches (110*156*260MM), KA3003x 7.7 Lbs (3.5Kg), KA3005x 9.5 Lbs (4.3Kg) | | |

Distributed Exclusively by SRA Soldering Products in the USA

Figure C.3: Korad Power Supply Datasheet.

# Appendix D

# Signal Conditioning Box

### D.0.1 5VDC Linear Regulating Circuit

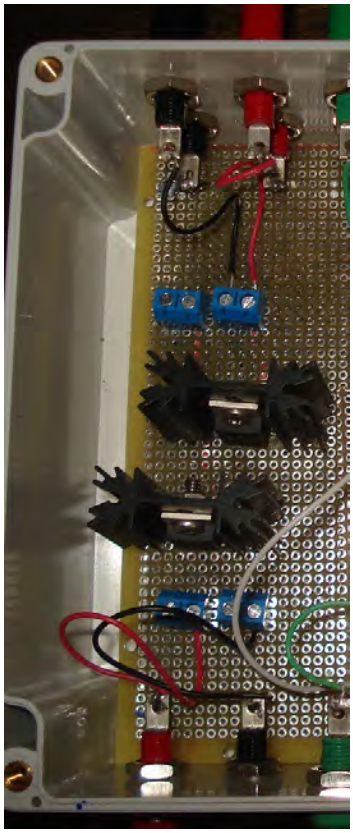


Figure D.1: 5VDC linear regulatory circuit.

### D.0.2 5VDC Linear Regulating Diagram

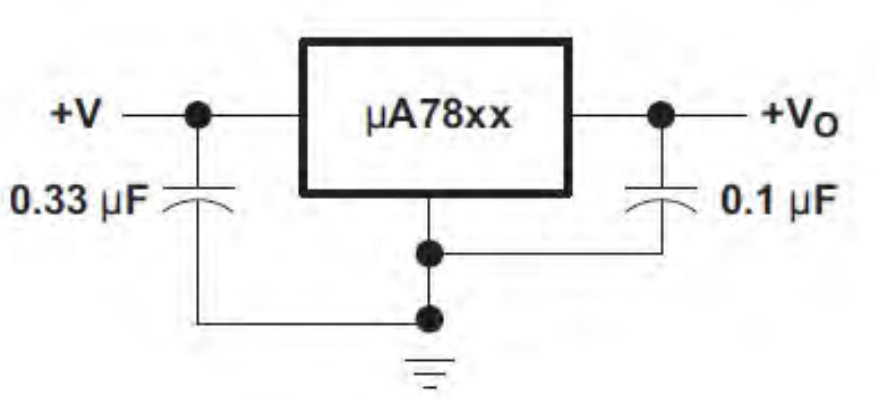



Figure D.2: 5VDC linear regulatory diagram.

### D.0.3 R/C Low Pass Filter Circuit

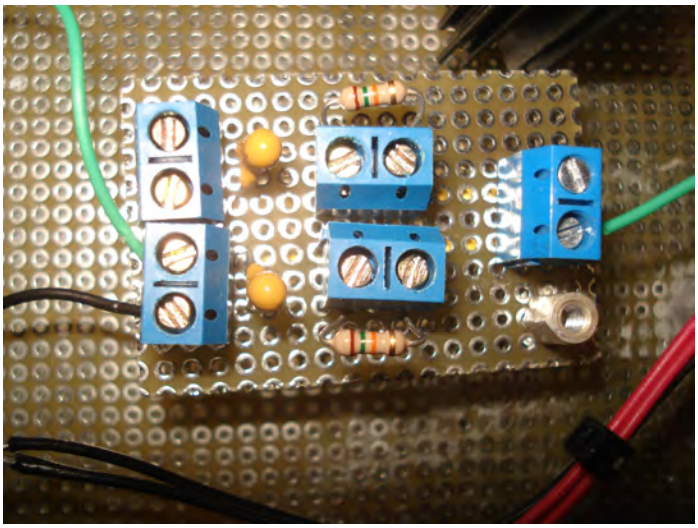


Figure D.3: R/C Low-Pass filter circuit.

### D.0.4 R/C Low Pass Filter Diagram

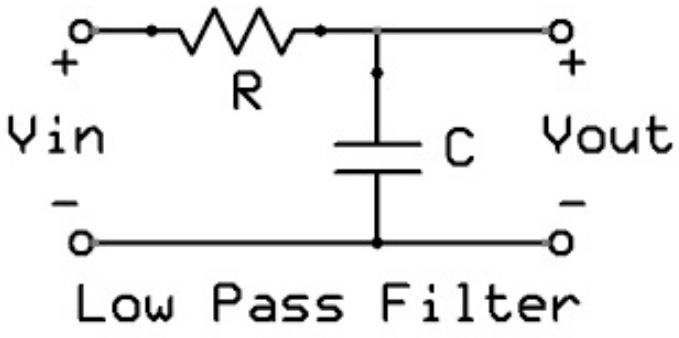



Figure D.4: R/C Low-Pass filter diagram.

### D.0.5 Solid-State Relay Circuit

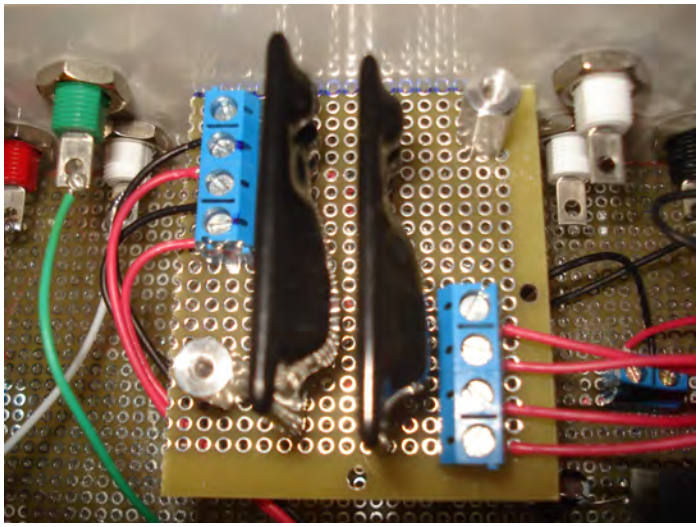


Figure D.5: Solid-State Relays circuit.

### D.0.6 Solid-State Relay Diagram

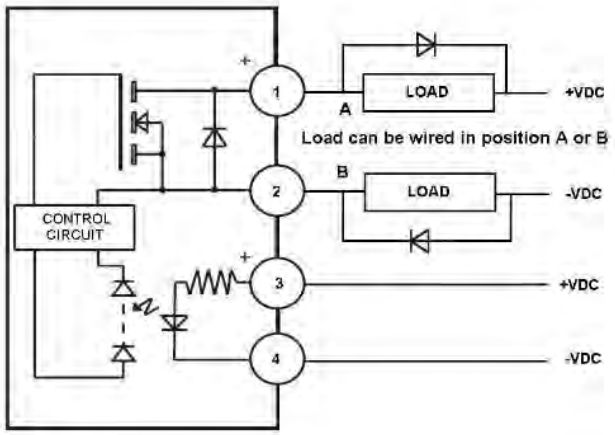



Figure D.6: Solid-State Relays diagram.

### D.0.7 Power Amplfiers Circuit

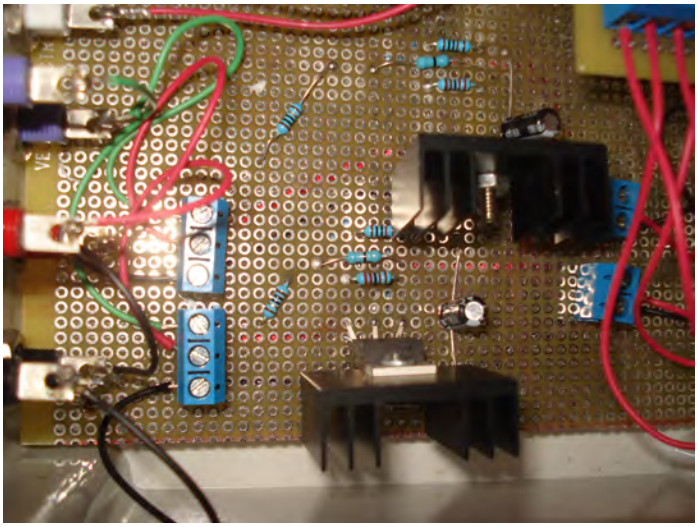


Figure D.7: Power Amplifiers circuit.

### D.0.8 Power Amplifier Diagram

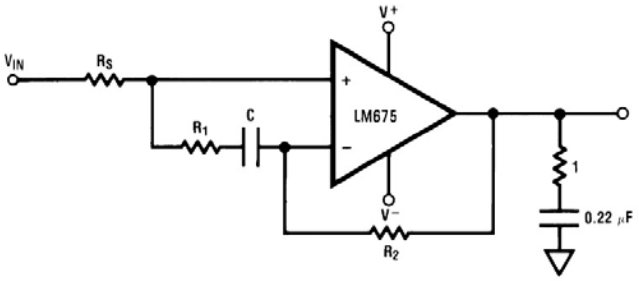



Figure D.8: Power Amplifiers diagram.

# Appendix E

# Calibration Script

### E.0.1 Rangefinder Calibration Script

```
%Clear the screen and being diary file
tic;
clc;
diary Points_5m_Nofilter_Final.txt;

excel_data1 = {};
excel_data2 = {};
excel_data3 = {};
mean_display1 = {};

% # of points to be taken
data_points = [0:1:100]; %#ok<*NBRAK>

% # readings per point
readings = 5;
count = 1;
count1 = 1;

%Open a figure for plotting points
figure;
```

```
for i=1:1:max(size(data_points)),

    uiwait(msgbox(['Change position to read: ' ...
        num2str(data_points(count1))]));

    %Take (5) readings for each distance
    for j=1:readings,

        %Pause for 5 seconds between readings
        pause(5);

        %Left Side
        %Get the value of the display block (raw value)
        blockhandle1 = ...
        getSimulinkBlockHandle('arduinomega2560_RangeFinder_Calibration/Display2');
        rto1 = get_param(blockhandle1,'RuntimeObject');
        a = rto1.InputPort(1).Data;

        %Store display block value for averaging
        mean_display1{j} = a;

        %Store values and points for analysis
        excel_data1(count, 1)  = {data_points(count1)};
        excel_data1(count, 2)  = {num2str(a, '% .4f')};

        %Increment display count
        count = count + 1;

    end
    disp(['Position ' num2str(data_points(count1))]);
    disp(['Mean: ' num2str(mean(cell2mat(mean_display1)))]);
    disp(['Min:  ' num2str(min(cell2mat(mean_display1)))]);
    disp(['Max:  ' num2str(max(cell2mat(mean_display1)))]);
    disp('------------------');

    %Store mean values for calibration
```

```
    excel_data2(count1,1) = {data_points(count1)}; %#ok<*SAGROW>
    excel_data2(count1,2) = {mean(cell2mat(mean_display1))};

    %Store Min and Max values for calibration
    excel_data3(count1,1) = {data_points(count1)};
    excel_data3(count1,2) = {min(cell2mat(mean_display1))};
    excel_data3(count1,3) = {max(cell2mat(mean_display1))};


    %Graph the points
    hold on; plot(count1-1, mean(cell2mat(mean_display1)), 'b*');

    %Increment store count
    count1 = count1+1;

    %Clear memory
    clear mean_display1;

end

diary off;
close(gcf);

%Write all (5) raw values to excel
save_path = 'C:\Joe\Thesis_Prelim\Laser Range ...
   Finder\IR_Sensor_Calibration_auto_5m_Nofilter_Final';
xlswrite(save_path, {'Distance (inches)', 'Values'}, 'Raw', 'A1');
xlswrite(save_path, excel_data1, 'Raw', 'A3');

%Write the mean value for the (5) samples taken at each distance
save_path = 'C:\Joe\Thesis_Prelim\Laser Range ...
   Finder\IR_Sensor_Calibration_auto_5m_Nofilter_Final';
xlswrite(save_path, {'Distance (inches)', 'Means'}, 'Mean', 'A1');
xlswrite(save_path, excel_data2, 'Mean', 'A3');

%Write the max value for the (5) samples taken at each distance
```

```
save_path = 'C:\Joe\Thesis_Prelim\Laser Range ...
    Finder\IR_Sensor_Calibration_auto_5m_Nofilter_Final';
xlswrite(save_path, {'Distance (inches)', 'Min', 'Max'}, 'MIN MAX', ...
    'A1');
xlswrite(save_path, excel_data3, 'MIN MAX', 'A3');

figure; plot(cell2mat(excel_data2(:,2)), 'r');

toc;
```

# Appendix F

# Low Fidelity Signal Model Explained

### F.0.1 Low Fidelity Signal Model - Full

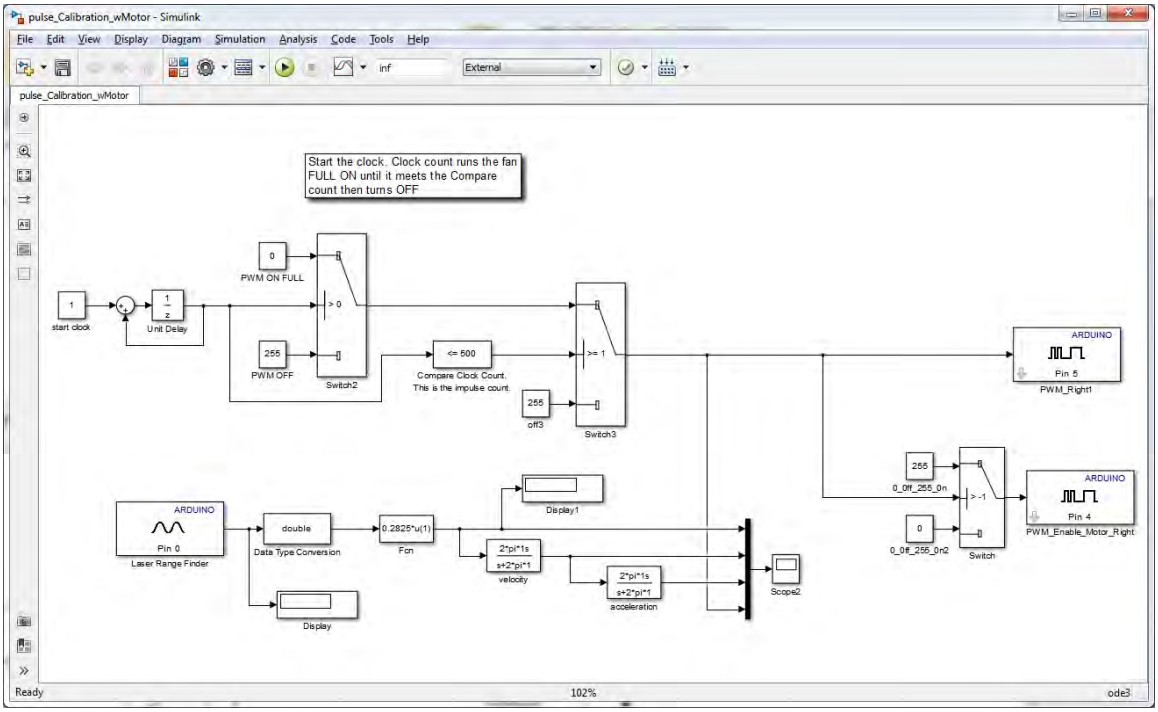



Figure F.1: Low Fidelity Signal Model.

Below are sections of the Signal Model explainations. Figure F.2 illustrates the command signal, Figure F.3 explains the position sensor I/O and Figure F.4 clarifies the PWM command to the fan and solid state relay.

### F.0.2 Low Fidelity Signal Model - 1

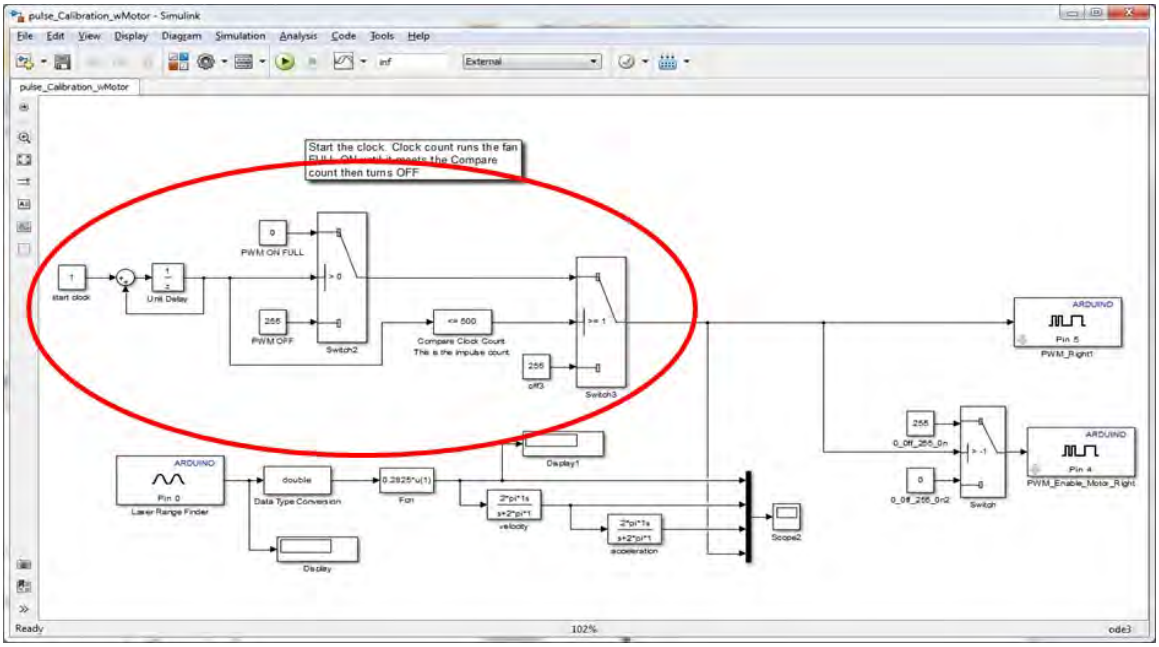



Figure F.2: Low Fidelity Signal Model - 1.

The model is setup to run at 100Hz (1/96). The "Start Clock" constant block is used to start the counter. A unit delay block combined with a summing block will count the number of clock tics. This value is linked to a compare block set to run for 500 counts (i.e. 5 seconds). The combination of switch blocks are used set the value of the PWM signal to the Arduino. When the "Start Clock" is initiated to 1, the count starts and a PWM of 0 is initiated. This event will command 24VDC be sent to the right fan. Once the counter reaches 500 the switch block will activate and a command of 255 will be sent to the Arduino PWM which will inturn cause 0VDC to be sent to the right fan to shut it off.

### F.0.3 Low Fidelity Signal Model - 2

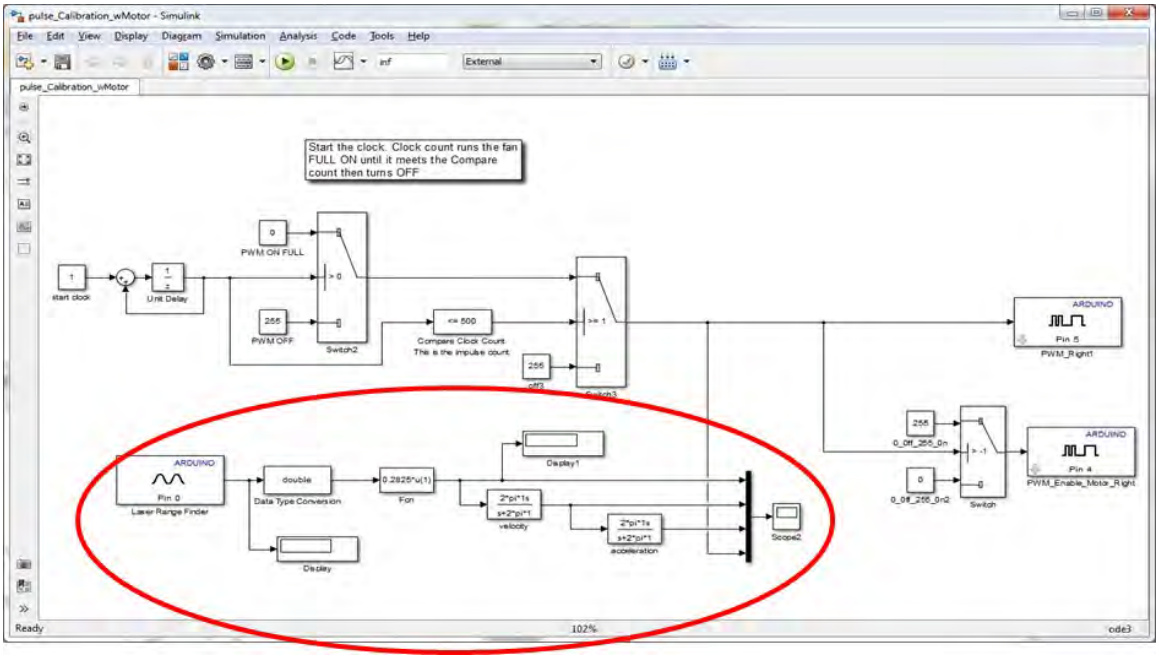



Figure F.3: Low Fidelity Signal Model - 2.

The Arduino analog input block is used to measure the analog signal returning from the position sensor. The block represents the voltage as a digital value (0-1023, minimum to maximum) from the rangefinder. This signal is sent thru a conversion block which turns the value into double percision representation and is passed to the function block which contains the linear equation obtained from the system calibration. The signal is tapped into from a derivative block which converts the position to velocity and acceleration. The signals are routed through a bus and out to a scope block which saves the data in the MATLAB workspace for data analysis.

### F.0.4 Low Fidelity Signal Model - 3

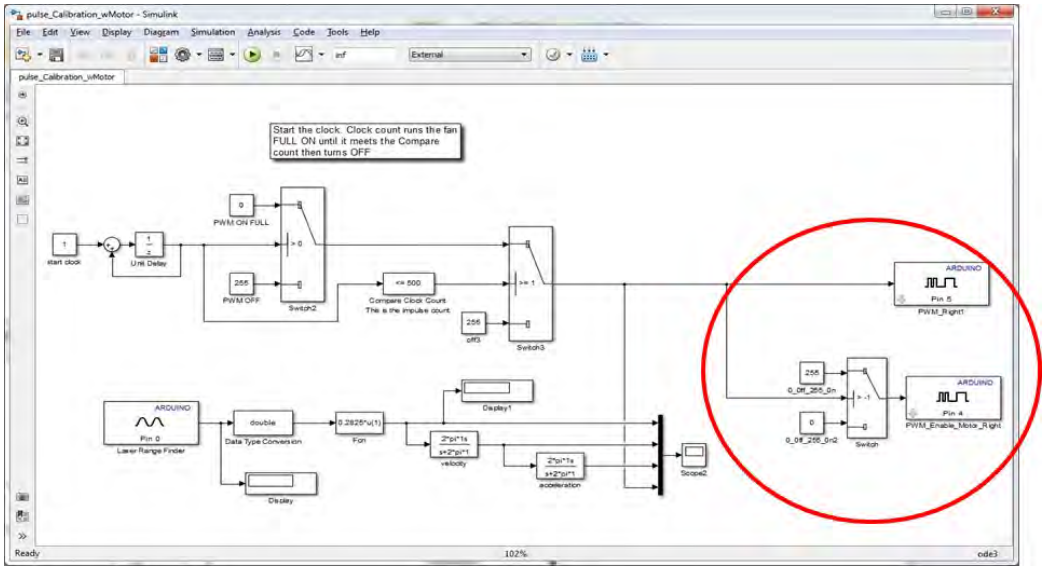



Figure F.4: Low Fidelity Signal Model - 3.

The PWM signal used to command voltage to the right fan comes from the switch block. Although the PWM signal has a range of 0-255, for preliminary signal testing the commands were hard coded to send either 0 ( for FULL on) or 255 (off). The commanded signal is tapped and sends a PWM command to both the fan output and the solid state relay for the right fan. The command to the solid state relay will close the connection and permit voltage to pass to the right fan.

### F.0.5 Low Fidelity Model Plotting Script

```
%Assign variables from workspace vector
t       = Step_Command.time;
pos     = Step_Command.signals.values(:,1);
vel     = Step_Command.signals.values(:,2);
accel   = Step_Command.signals.values(:,3);
pwm     = Step_Command.signals.values(:,4);

%%%%%%%%%%%%%%%%%%%%%%%%
%Plot cart data - All
figure;
stairs(t,pos,'r');
hold on; stairs(t,vel,'b');
hold on; stairs(t,accel,'g');
hold on; stairs(t,pwm/10,'k');
grid on;
title('POD Step Response');
legend('position', 'velocity', 'acceleration', 'PWM');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Plot cart data - Position Only
figure;
stairs(t,pos,'r');
grid on;
title('POD Stationary');
legend('position');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Plot cart data - Position and Velocity Only
figure;
stairs(t,pos,'r');
grid on;
title('POD Stationary');
legend('position');
```

# Appendix G

# Closed Loop Hyperloop MATLAB Code

### G.0.1 Fuzzy Logic Code

```
function y = fcn(x1, x2, x3)
%#codegen
y = 0;

%1 if the POD has crossed the
%desired deceleration point
decel = x3(1);

%Determines direction
%to send the POD
direction = x3(2);


% y is the value of the output between -1 and 1
% x1 is the value to test for input 1
% x2 is the value to test for input 2
% x3(1) is the value to turn the fuzzy control logic on
% x3(2) is the value to determine which fan is accelerating
% the pod, prior to the controller becoming active
% c1 is an 1xn array of the fuzzy rule set center points for input 1
% c2 is an 1xn array of the fuzzy rule set center points for input 2
```

```
% rules is a n x m matrix where n is the length of c1 and
% m is the length of c2

%Time to engage the controller
if decel

    %input 1 fuzzy set (e)
    c1 = [-1 -.4 -.3 0 .3 .4 1];

    %input 2 fuzzy set (ce)
    c2 = [-1 -.4 -.2 0 .2 .4 1];

    %output fuzzy set
    u = [-1 -.8 -.5 0 .5 .8 1];

    nb = u(1);
    nm = u(2);
    ns = u(3);
    z = u(4);
    ps = u(5);
    pm = u(6);
    pb = u(7);

    rules = [nb pb pb pb pb pb pb;
             nb z ps pm pb pb pb;
             nb ns z ps pm pb pb;
             nb nm ns z ps pm pb;
             nb nm nm ns z ps pb;
             nb nb nm nm ns z pb;
             nb nb nb nb nb ns pb];

    %determine membership
    mem1 = fuzzify(x1,c1);
    mem2 = fuzzify(x2,c2);

    %find non zero elements
```

```
    nz1 = find(mem1 > 0);
    nz2 = find(mem2 > 0);

    % check to see if either membership value only has one element
    if length(nz1) < 2 || length(nz2) < 2
        if length(nz1) < 2 && length(nz2) >= 2
            m = [mem2(nz2(1)) mem2(nz2(2))];
            RV = [rules(nz1(1),nz2(1)) rules(nz1(1),nz2(2))];
            y = RV*m';
        elseif length(nz2) < 2 && length(nz1) >= 2
            m = [mem1(nz1(1)) mem1(nz1(2))];
            RV = [rules(nz1(1),nz2(1)); rules(nz1(2),nz2(1))];
            y = m*RV;
        elseif length(nz1) < 2 && length(nz2) < 2
            y = mem1(nz1(1))*rules(nz1(1),nz2(1))*mem2(nz2(1));
        end

        % otherwise calculate only the nonzero elements
    else
        m1 = [mem1(nz1(1)) mem1(nz1(2))];
        m2 = [mem2(nz2(1)) mem2(nz2(2))];
        RV = [rules(nz1(1),nz2(1)) rules(nz1(1),nz2(2));
            rules(nz1(2),nz2(1)) rules(nz1(2),nz2(2))];
        y = m1*RV*m2';

    end
else
    %turn the right fan on to accelerate
    if direction > 0
        y = 1;
    %turn the left fan on to accelerate
    else
        y = -1;
    end
end
```

### G.0.2 Fuzzy Logic Code

```
function y = fuzzify(x, centers)
% y = fuzzify(x, centers)
% y is an 1xn array of the membership in each fuzzy rule
% where n is the number of centers
% x is the value to test
% centers is an 1xn array of the fuzzy rule set center points

%Initialize the output vector y
y = zeros(1,max(size(centers)));

% Check to see if we are outside the universe of discourse
if x>= centers(max(size(centers)))
    y(max(size(centers))) = 1;
elseif x<= centers(1)
    y(1) = 1;
end

% check membership in each fuzzy set
for i=1:max(size(centers))-1
    if x <= centers(i+1) && x > centers(i)
        y(i) = (x-centers(i+1))/(centers(i)-centers(i+1));
        y(i+1) = (x-centers(i))/(centers(i+1)-centers(i));
    end
end
```

### G.0.3 Send the pod to the right

```
%%%%%%%%%%%%%%%%%%%%%%%%
% Left to Right command

%Define the destination (inches)
% dest = 10;
dest = 7;

%Define where to active the controller
% decel = 78;
decel = 77;

%Define max voltage supply to the fan
voltage = 24;

%Set the desired voltage
set_param('Hyperloop_Fuzzy2/Fan_Gain', 'Gain', num2str(voltage));

%Set the desired destination
set_param('Hyperloop_Fuzzy2/Destination_Position', 'Value', ...
    num2str(dest));

%Set the desired decel set point
set_param('Hyperloop_Fuzzy2/Decel_SetPoint_Cmd', 'Value', ...
    num2str(decel));

%Start the Simulation model
set_param('Hyperloop_Fuzzy2','SimulationCommand', 'start');

pause(2);

%Begin the simulation
set_param('Hyperloop_Fuzzy2/Start', 'Value', '1');
```

```
%Run the script for 10 seconds
%This should be plenty of time
pause(10);

%Stop the fans
set_param('Hyperloop_Fuzzy2/Start', 'Value', '0');

%Shut down the simulation model
set_param('Hyperloop_Fuzzy2','SimulationCommand', 'stop');

%Pause while data is saved to
%to the workspace
pause(2);

%Plot the saved workspace data
t = Stop_Eval(:,1);
RFV = Stop_Eval(:,2);
LFV = Stop_Eval(:,3);
Ppos = Stop_Eval(:,4);
Pvel = Stop_Eval(:,5);
PPvel = Stop_Eval(:,6);

min_pos = min(Ppos);
temp = find(Stop_Eval(:,4) == min_pos);
time_min_pos = t(temp(1));

figure;
stairs(t, RFV, 'r');
hold on; stairs(t, LFV, 'b');
hold on; stairs(t, Ppos, 'g');
hold on; stairs(t, Pvel, 'k');
hold on; stairs(t, PPvel, 'c--');

%Show the desired target
line([0 12],[dest dest],'Color','c','LineWidth',2)
grid on;
```

```
axis([0 12 -1 100]);
title(['Min Position: ' num2str(min_pos) '   Time: ' ...
    num2str(time_min_pos)])
legend('RFv', 'LFv', 'Ppos', 'Pvel', 'PPvel');
```

### G.0.4 Send the pod to the left

```
%%%%%%%%%%%%%%%%%%%%%%%%
% Right to left command

%Define the destination
% dest = 85;
dest = 88;

%Define where to activate the controller
% decel = 16.5;
decel = 17.5;

%Define max voltage supply to the fan
voltage = 24;

%Set the desired voltage
set_param('Hyperloop_Fuzzy2/Fan_Gain', 'Gain', num2str(voltage));

%Set the desired destination
set_param('Hyperloop_Fuzzy2/Destination_Position', 'Value', ...
    num2str(dest));

%Set the desired decel set point
set_param('Hyperloop_Fuzzy2/Decel_SetPoint_Cmd', 'Value', ...
    num2str(decel));

%Start the Simulation Model
set_param('Hyperloop_Fuzzy2','SimulationCommand', 'start');

pause(2);

%Begin the simulation
set_param('Hyperloop_Fuzzy2/Start', 'Value', '1');
```

```
%Run the script for 10 seconds
%This should be plenty of time
pause(10);

%Stop the fans
set_param('Hyperloop_Fuzzy2/Start', 'Value', '0');

%Shut down the simulation model
set_param('Hyperloop_Fuzzy2','SimulationCommand', 'stop');

%Pause while data is saved to
%to the workspace
pause(2);

%Plot the saved workspace data
t = Stop_Eval(:,1);
RFV = Stop_Eval(:,2);
LFV = Stop_Eval(:,3);
Ppos = Stop_Eval(:,4);
Pvel = Stop_Eval(:,5);
PPvel = Stop_Eval(:,6);

max_pos = max(Ppos);
temp = find(Stop_Eval(:,4) == max_pos);
time_max_pos = t(temp(1));

figure;
stairs(t, RFV, 'r');
hold on; stairs(t, LFV, 'b');
hold on; stairs(t, Ppos, 'g');
hold on; stairs(t, Pvel, 'k');
hold on; stairs(t, PPvel, 'c--');

%Show the desired target
line([0 12],[dest dest],'Color','c','LineWidth',2)
grid on;
```

```
axis([0 12 -1 100]);
title(['Max Position: ' num2str(max_pos) '   Time: ' ...
    num2str(time_max_pos)])
legend('RFv', 'LFv', 'Ppos', 'Pvel', 'PPvel');
```

### G.0.5 Left to Right Analysis Script

```
%Right To Left Analysis
commanded_position = 10;

%Pull values from saved data
t = Stop_Eval(:,1);
RFV = Stop_Eval(:,2);
LFV = Stop_Eval(:,3);
Ppos = Stop_Eval(:,4);
Pvel = Stop_Eval(:,5);
PPvel = Stop_Eval(:,6);

%Find MAX values
min_pos = min(Ppos);
temp = find(Stop_Eval(:,4) == min_pos);
time_min_pos = t(temp(1));

%Find Right Fan values
right_fan = find(RFV ~= 0);
right_fan_on = t(right_fan(1));
right_fan_off = t(right_fan(end));
right_fan_on_total = right_fan_off - right_fan_on;

%Find Left Fan values
left_fan = find(LFV ~= 0);
left_fan_on = t(left_fan(1));
left_fan_off = t(left_fan(end));
left_fan_on_total = left_fan_off - left_fan_on;

%Calculate Over/Undershoot
over_under_shoot = commanded_position - min_pos;

%Find the MAX velocity after the Right Fan is enabled
velocity_max = max(Pvel(right_fan(1):end));
```

```
temp1 = find(Pvel == velocity_max);
time_max_vel = t(temp1(1));

%Save data
excel_data = {};

excel_data(1, 1)  = {'Right Fan On'};
excel_data(1, 2)  = {'Right Fan Off'};
excel_data(1, 3)  = {'Right Fan Total Time'};
excel_data(2, 1)  = {right_fan_on};
excel_data(2, 2)  = {right_fan_off};
excel_data(2, 3)  = {right_fan_on_total};

excel_data(4, 1)  = {'Left Fan On'};
excel_data(4, 2)  = {'Left Fan Off'};
excel_data(4, 3)  = {'Left Fan Total Time'};
excel_data(5, 1)  = {left_fan_on};
excel_data(5, 2)  = {left_fan_off};
excel_data(5, 3)  = {left_fan_on_total};

excel_data(7, 1)  = {'Max. Velocity'};
excel_data(7, 2)  = {'Max. Velocity Time'};
excel_data(8, 1)  = {velocity_max};
excel_data(8, 2)  = {time_max_vel};

excel_data(10, 1)  = {'Min. Position'};
excel_data(10, 2)  = {'Min. Position Time'};
excel_data(10, 3)  = {'Beyond Target Position'};
excel_data(11, 1)  = {min_pos};
excel_data(11, 2)  = {time_min_pos};
excel_data(11, 3)  = {over_under_shoot};

%Write Data to EXCEL
save_path = [pwd '\Left2Right_2016_' datestr(now, 'HH') '_' ...
   datestr(now, 'MM')];
xlswrite(save_path, excel_data, 'Analysis', 'A1');
```

### G.0.6 Right to Left Analysis Script

```
%Right To Left Analysis
commanded_position = 85;

%Pull values from saved data
t = Stop_Eval(:,1);
RFV = Stop_Eval(:,2);
LFV = Stop_Eval(:,3);
Ppos = Stop_Eval(:,4);
Pvel = Stop_Eval(:,5);
PPvel = Stop_Eval(:,6);

%Find MAX values
max_pos = max(Ppos);
temp = find(Stop_Eval(:,4) == max_pos);
time_max_pos = t(temp(1));

%Find Right Fan values
right_fan = find(RFV ~= 0);
right_fan_on = t(right_fan(1));
right_fan_off = t(right_fan(end));
right_fan_on_total = right_fan_off - right_fan_on;

%Find Left Fan values
left_fan = find(LFV ~= 0);
left_fan_on = t(left_fan(1));
left_fan_off = t(left_fan(end));
left_fan_on_total = left_fan_off - left_fan_on;

%Calculate Over/Undershoot
over_under_shoot = max_pos - commanded_position;

%Find the MAX velocity after the Right Fan is enabled
velocity_max = max(Pvel(right_fan(1):end));
```

```
temp1 = find(Pvel == velocity_max);
time_max_vel = t(temp1(1));

%Save data
excel_data = {};

excel_data(1, 1)  = {'Right Fan On'};
excel_data(1, 2)  = {'Right Fan Off'};
excel_data(1, 3)  = {'Right Fan Total Time'};
excel_data(2, 1)  = {right_fan_on};
excel_data(2, 2)  = {right_fan_off};
excel_data(2, 3)  = {right_fan_on_total};

excel_data(4, 1)  = {'Left Fan On'};
excel_data(4, 2)  = {'Left Fan Off'};
excel_data(4, 3)  = {'Left Fan Total Time'};
excel_data(5, 1)  = {left_fan_on};
excel_data(5, 2)  = {left_fan_off};
excel_data(5, 3)  = {left_fan_on_total};

excel_data(7, 1)  = {'Max. Velocity'};
excel_data(7, 2)  = {'Max. Velocity Time'};
excel_data(8, 1)  = {velocity_max};
excel_data(8, 2)  = {time_max_vel};

excel_data(10, 1)  = {'Max. Position'};
excel_data(10, 2)  = {'Max. Position Time'};
excel_data(10, 3)  = {'Beyond Target Position'};
excel_data(11, 1)  = {max_pos};
excel_data(11, 2)  = {time_max_pos};
excel_data(11, 3)  = {over_under_shoot};

%Write Data to EXCEL
save_path = [pwd '\Right2Left_2016_' datestr(now, 'HH') '_' ...
   datestr(now, 'MM')];
xlswrite(save_path, excel_data, 'Analysis', 'A1');
```